ซีพีเอ็ม

CP/M OPERATING SYSTEM

ยืน ภู่วรวรรณ
พิชิต สุขเจริญพงษ์
สมนึก คีรีโต



ซีพีเอ็ม
CP/M OPERATING SYSTEM

ยืน ภู่วรวรรณ
พิชิต สุขเจริญพงษ์
สมนึก คีรีโต

บริษัท ซีเอ็ดยูเคชั่น จำกัด
SCIENCE, ENGINEERING & EDUCATION CO.,LTD.

ราคา 58 บาท

ISBN 974-7965-39-9
พิมพ์ครั้งที่ 1 พ.ศ. 2527

จัดพิมพ์และจัดจำหน่ายโดย
บริษัท ซีเอ็ดยูเคชั่น จำกัด
SCIENCE, ENGINEERING & EDUCATION CO.,LTD.
800/43-45 ซอยตระกูลสุข ถนนอโศกดินแดง
เขตห้วยขวาง กท. 10400 โทร.245-0390

พิมพ์ที่ หจก. เอช-เอน การพิมพ์
402/16 ซอยดอนกุศล เขตยานนาวา กทม. 10120
นายประพันธ์ จิระมงคล ผู้พิมพ์ผู้โฆษณา พ.ศ. 2527

คำนำ

ปัจจุบันไมโครคอมพิวเตอร์ขนาดเล็กที่ใช้กันในราคาเพียงสอง-สามหมื่นบาทมีขีดความสามารถสูงกว่าคอมพิวเตอร์เมื่อสิบกว่าปีก่อนที่มีราคาเป็นล้าน ความแตกต่างระหว่างเมนเฟรมคอมพิวเตอร์กับไมโครคอมพิวเตอร์ที่เป็นประโยชน์ต่อผู้ใช้คือ เรื่องของซอฟต์แวร์ ราคาของซอฟต์แวร์ในระดับเมนเฟรมคอมพิวเตอร์มีราคาสูงมาก เพราะผู้สร้างซอฟต์แวร์มีตลาดการใช้ไม่มากนัก แต่ซอฟต์แวร์ของไมโครคอมพิวเตอร์มีราคาถูก เพราะมีผู้ใช้กันมาก ผู้ผลิตซอฟต์แวร์สามารถสร้างซอฟต์แวร์เพื่อขายในตลาดทั่วไปได้ และด้วยเหตุนี้เองตลาดซอฟต์แวร์ในระดับเครื่องไมโครคอมพิวเตอร์จึงกว้างขวางอย่างมาก

สาเหตุหนึ่งที่ทำให้ซอฟต์แวร์ประยุกต์ในระดับไมโครคอมพิวเตอร์เป็นมาตรฐานใช้งานได้กว้างขวาง เห็นจะได้แก่ระบบโปรแกรมจัดระบบงาน (operating system) เพราะไมโครคอมพิวเตอร์ส่วนใหญ่ไม่ว่าจะเป็นยี่ห้ออะไรจะใช้โปรแกรมจัดระบบงานเหมือนกัน จึงทำให้ซอฟต์แวร์ประยุกต์ใช้ได้กว้างขวาง

ในบรรดาโปรแกรมจัดระบบงานที่แพร่หลายมากที่สุดเห็นจะได้แก่ ซีพีเอ็ม (CP/M) ที่ออกแบบและสร้างโดยบริษัทดิจิตอลรีเสิร์จ ซึ่งปัจจุบันเครื่องคอมพิวเตอร์เกือบทุกยี่ห้อใช้โปรแกรมจัดระบบงานนี้ จึงทำให้ผู้ผลิตซอฟต์แวร์ประยุกต์ต่าง ๆ พยายามสร้างโปรแกรมสำเร็จรูปบน CP/M

โปรแกรมที่จะใช้ได้บน CP/M มีมากกว่า 10,000 โปรแกรม ซึ่งไม่ว่าจะเป็นโปรแกรมภาษาเช่น เบสิค ฟอร์แทรน โคบอล พาสคาล ฟอร์ท ลิฟท์ อาดา อาร์พีจี พีแอลวัน ฯลฯ นอกจากนี้โปรแกรมจำพวกเวิร์ดโปรเซสซิ่ง อิเล็กทรอนิกส์เวอร์คชีตระบบฐานข้อมูล ฯลฯ เมื่อเป็นเช่นนี้ CP/M จึงเป็นเรื่องที่ผู้ใช้ไมโครคอมพิวเตอร์ทั่วไปรู้จักอย่างดี

หนังสือ CP/M เล่มนี้จึงมีจุดมุ่งหมายที่จะทำให้ผู้ใช้ไมโครคอมพิวเตอร์ทั่วไปได้เข้าใจในโปรแกรมจัดระบบงานนี้มากขึ้น เพราะภายในโปรแกรมจัดระบบงานนี้ยังมียูทิลิตี้อื่น ๆ ประกอบอีกมาก เช่น แอสเซมเบลอร์ เอดิเตอร์ และดีบักเกอร์ ซึ่งสิ่งที่ประกอบมานี้จะช่วยผู้ใช้ไมโครคอมพิวเตอร์จัดการระบบตามความต้องการได้ แต่อย่างไรก็ตามเทคนิคและวิธีการใช้โปรแกรมยูทิลิตี้เหล่านี้จำเป็นต้องขยายความเพื่อให้ผู้ใช้ได้เข้าใจได้ง่าย

กลุ่มผู้เรียบเรียงมีความต้องการที่จะให้หนังสือ CP/M นี้เป็นคู่มือการใช้ไมโครคอมพิวเตอร์สำหรับผู้สนใจไมโครคอมพิวเตอร์ เพื่อสร้างพื้นฐานความรู้ในการใช้ไมโครคอมพิวเตอร์ให้มีประสิทธิภาพดียิ่งขึ้น ดังนั้นเนื้อหาในตอนแรก คือบทที่ 1 ถึง 9 เป็นเรื่องเกี่ยวกับพื้นฐาน ส่วนในบทที่ 10 เป็นบทที่

ว่าด้วยโครงสร้างของ CP/M เพื่อให้ผู้ที่ต้องการพัฒนาแก้ไขหรือเขียนโปรแกรมที่จะต้องทำงานรวมกับ CP/M ได้เข้าใจหลักการและวิธีการเรียกใช้

ส่วนในภาคผนวกได้เน้นในเรื่องระบบซอฟต์แวร์ที่ใช้บน CP/M ที่น่าสนใจซึ่งหลายโปรแกรมเป็นโปรแกรมทางภาษา เช่น ฟอร์แทรน โคบอล ส่วนในภาคผนวกสุดท้ายว่าด้วยการจัดแผ่นดิสค์ เพื่อให้เข้าใจหลักการจัดการระบบดิสเก็ตต์ของ CP/M

กลุ่มผู้เรียบเรียงคาดหวังว่า หนังสือ CP/M เล่มนี้คงมีส่วนให้ผู้อ่านได้เข้าใจในระบบไมโครคอมพิวเตอร์มากยิ่งขึ้น หากมีข้อบกพร่องประการใด ผู้เขียนขอน้อมรับคำแนะนำด้วยความขอบคุณยิ่ง

ยืน ภู่วรวรรณ

ดร.พิชิต สุขเจริญพงษ์

สมนึก คีรีโต

ห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์

มหาวิทยาลัยเกษตรศาสตร์

สารบัญ

1

รู้จักกับ CP/M

ประวัติความเป็นมาของคอมพิวเตอร์

เมื่อคอมพิวเตอร์อุบัติขึ้นมาบนโลกโดยมีนักวิทยาศาสตร์และวิศวกรเป็นจำนวนมากเป็นผู้คิดและประดิษฐ์ขึ้น คอมพิวเตอร์ก็ค่อย ๆ เติบใหญ่ขึ้นมาทุกวัน การวิวัฒนาการทางคอมพิวเตอร์นั้นไปในแนวทางสองแขนงคือ ทางฮาร์ดแวร์และซอฟต์แวร์ ฮาร์ดแวร์คือรูปโครงภายนอก ซึ่งมีการเกิดและตายไปแล้วจวบจนในปัจจุบันอยู่ในรุ่นระดับที่ 4 คอมพิวเตอร์รุ่นแรกก็คือคอมพิวเตอร์ยุคหลอดสุญญากาศ รุ่นที่สองก็คือทรานซิสเตอร์ รุ่นระดับ 3 ก็คือไอซี รุ่นปัจจุบันคือ แอลเอสไอ(LSI) และรุ่นที่ 5 ต่อไปคือ วีแอลเอสไอ (VLSI-very large scale integrated circuit) การเติบโตแตกลูกหลานนี้เป็นทางฮาร์ดแวร์

หากคอมพิวเตอร์ที่สร้างมามีแต่โครงร่างทางฮาร์ดแวร์ คอมพิวเตอร์ก็ไม่ผิดอะไรกับเครื่องจักร เช่น รถยนต์ รถไฟ ที่เมื่อมีแรงขับดันก็วิ่งได้ ถ้ามีอะไรขวางหน้าก็ชนโดยไม่คิดอะไรทั้งสิ้น คอมพิวเตอร์จึงต้องอาศัยซอฟต์แวร์เพื่อให้มีการทำงานได้

การพัฒนาทางซอฟต์แวร์ก็เช่นเดียวกันได้เจริญก้าวหน้าเติบใหญ่ขึ้นเป็นลำดับ หรือหากจะพูดถึงระดับสติปัญญาของคอมพิวเตอร์ ๆ ก็มีระดับสมองค่อย ๆ โตขึ้นเรื่อย ๆ การชาญฉลาดของคอมพิวเตอร์ก็ล้วนแล้วแต่ถ่ายทอดแนวความคิดต่าง ๆ ของมนุษย์ไปทั้งสิ้น

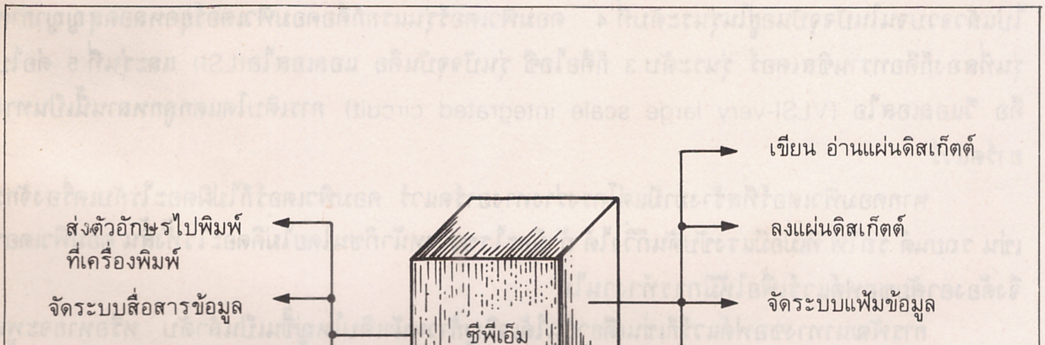
การสร้างสติปัญญาให้กับคอมพิวเตอร์จำต้องมีการสะสมเพิ่มพูนความรู้ขึ้นเรื่อย ๆ และเนื่องจากต้องมีการสะสมความรู้นี้เองจึงมีผู้พยายามสร้างโปรแกรมหรือซอฟต์แวร์พื้นฐานให้กับคอมพิวเตอร์ในการจัดการกับระบบเสียก่อน หากระบบการจัดการกับตัวเองได้ดีแล้ว การควบคุมการทำงานต่าง ๆ ของคอมพิวเตอร์ต่อไปก็จะทำได้ง่าย ดังนั้นผู้ผลิตคอมพิวเตอร์จึงจำเป็นที่จะต้องพัฒนาให้โปรแกรมพื้นฐานนี้ให้เป็นที่ยอมรับและใช้งานได้ง่าย โปรแกรมดังกล่าวจึงมีผู้สนใจและอยากช่วยเสริมสร้างสติปัญญาต่อให้เครื่องต่อไป

โปรแกรมจัดระบบงานคืออะไร

เมื่อผู้เริ่มมีคอมพิวเตอร์ หรือเริ่มใช้ไมโครคอมพิวเตอร์ สิ่งที่พบเห็นอย่างหนึ่งคือ คำว่า DOS, CP/M, MSDOS หรืออะไรต่อมิอะไรที่เกี่ยวกับคำว่า OS OS หรือ Operating System หรือโปรแกรมจัดระบบงานเข้ามาเป็นส่วนพื้นฐานของเครื่องที่มีการใช้แผ่นจานแม่เหล็กทั้งสิ้น

คอมพิวเตอร์ไม่ว่าจะเป็นไมโครราคาเพียงสามหมื่น หรือเครื่องเมนเฟรมราคาเป็นล้านบาท ก็จำเป็นอย่างยิ่งที่จะต้องมีระบบซอฟต์แวร์อย่างใดอย่างหนึ่งที่ทำให้เครื่องสามารถทำงานได้ด้วยระบบที่มีประสิทธิภาพ

หน้าที่หลักของโปรแกรมจัดระบบงาน หรือที่เรียกว่า OS มีสามประการ กล่าวคือ ประการแรก เป็นตัวจัดระบบสิ่งแวดล้อมซึ่งได้แก่ อินพุต-เอาต์พุตให้มีระเบียบ ง่ายต่อผู้ใช้จะสั่งงาน อุปกรณ์อินพุต-เอาต์พุท ได้แก่ ซีอาร์ที คีย์บอร์ด เครื่องพิมพ์ ฟลอปปี้ดิสค์ ฯลฯ การจัดการในส่วนนี้ ได้แก่การส่งข้อความออกเครื่องพิมพ์ ออกซีอาร์ที การรับคีย์บอร์ด เป็นต้น ประการที่สองของหน้าที่ OS คือเป็นระบบที่ใช้ในการจัดการข้อมูลกับแผ่นจานแม่เหล็ก ทำหน้าที่จัดหมวดหมู่ การเก็บข้อมูลบนแผ่นจานแม่เหล็ก ทำหน้าที่ในการบอกสถานะของแผ่นจานแม่เหล็กว่ามีพื้นที่เหลือเท่าใดเหลือบริเวณใด ถ้ามีการเขียนอ่านจะเขียนอ่านในบริเวณใดได้อย่างถูกต้อง ประการที่สามเป็นตัวที่ใช้ทำหน้าที่ในการโหลดและเอ็กซีคิวโปแกรมของผู้สั่งงาน โดยผู้ใช้สามารถเขียนโปรแกรมอื่นได้อีกและให้ตัวโปรแกรมจัดระบบเป็นตัวสั่งโหลดลงมาบนพื้นที่หน่วยความจำ และทำการเอ็กซีคิวโปรแกรมได้นั่นเอง การจัดการในข้อนี้จึงอยู่ที่การพยายามใช้พื้นที่ของหน่วยความจำให้ได้ประสิทธิภาพสูงสุดเท่าที่จะเป็นไปได้



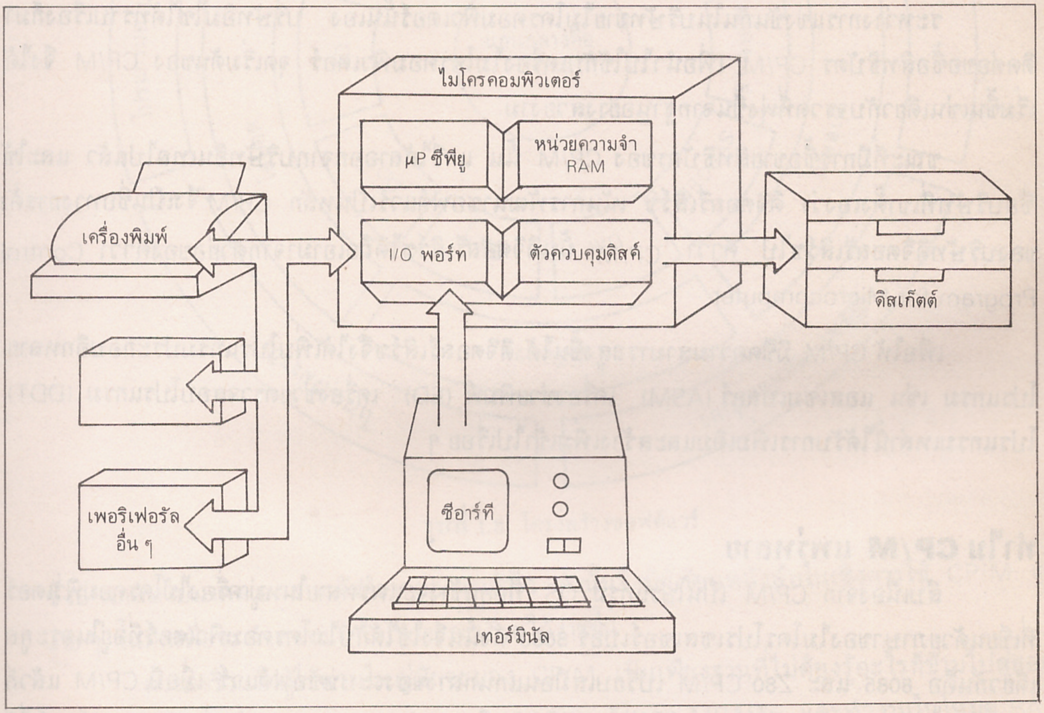
รูปที่ 1.1 หน้าที่การทำงานของ CP/M

ไมโครคอมพิวเตอร์คืออะไร

ไมโครคอมพิวเตอร์ได้แพร่หลายอย่างรวดเร็ว ซึ่งอาจจะด้วยเหตุที่เทคโนโลยีด้านนี้เจริญเติบโต และทำให้ไมโครคอมพิวเตอร์มีราคาถูกลงมาก ประจวบกับขีดความสามารถของไมโครคอมพิวเตอร์ก็สูงขึ้น

อย่างไรก็ตามจากการที่ไมโครคอมพิวเตอร์มีราคาไม่แพงนักผู้ออกแบบระบบซอฟต์แวร์จัดระบบการทำงานจึงมองเห็นในแงให้ผู้ใช้ได้เพียงคน ๆ เดียวและเครื่องทำงานได้ทีละงานเท่านั้น

โดยโครงสร้างของโปรแกรมจัดระบบงานจะเป็นตัวที่ยอมรับในอุปกรณ์อินพุตเอาต์พุตเพอริเฟอรัลต่าง ๆ โดยตัวโปรแกรม CP/M ซึ่งเป็น OS ที่แพร่หลายและนิยมที่สุดยอมรับการต่อโครงสร้างไมโครคอมพิวเตอร์ให้มี CPU และหน่วยความจำ สำหรับหน่วยความจำนั้น CP/M ยอมรับได้จาก 16K จนถึง 64 กิโลไบท์ นอกจากนี้ก็มี CRT มอนิเตอร์และคีย์บอร์ด เครื่องพิมพ์อินพุตเอาต์พุตพอร์ทอื่นและดิสเก็ตต์สำหรับดิสเก็ตต์นั้นยังสามารถเพิ่มขีดความสามารถให้มีดิสเก็ตต์ได้ถึง 4 ช่อง



รูปที่ 1.2 ระบบไมโครคอมพิวเตอร์

ประวัติความเป็นมาของ CP/M

ประมาณปี พ.ศ. 2513-2514 บริษัทอินเทลได้ประกาศความสำเร็จในการสร้างไมโครโปรเซสเซอร์เบอร์ 4004 ครั้นปี พ.ศ. 2516 อินเทลก็ประสบผลสำเร็จอีกครั้งด้วยการสร้างไมโครโปรเซสเซอร์เบอร์ 8080 ขึ้น ไมโครคอมพิวเตอร์ที่ใช้ซีพียูเบอร์ 8080 และออกมาสู่ท้องตลาดในปี พ.ศ. 2517 โดยบริษัท MITS (อัลแตร์) โดยขายออกมาในรูปคิทประกอบเอง มีราคาต่ำกว่า 1,000 เหรียญสหรัฐ และในเวลาไล่เลี่ยกันนั้นเองก็มีบริษัทผลิตไมโครคอมพิวเตอร์ออกขายแข่งขันกันในระยะแรกได้แก่ อิมไซ (Imsai) โครเมมโค (Cromemco) และนอร์ทสตาร์ (North Star) ในระยะแรกทั้งหมดหันมาใช้ระบบบัสที่เรียกว่า S-100 ซึ่งออกแบบโดยบริษัทอิมไซ

ขณะที่เกิดการแข่งขันกันนั้น วิศวกรของบริษัทอินเทลชื่อ แกรี่ คิลดอลล์ (Gary Kildall) ได้ทำงานเป็นที่ปรึกษาให้กับคนภายนอกด้วยการใช้ภาษาที่ชื่อว่า PL/M โดยภาษานี้อินเทลเป็นผู้สร้างขึ้นบนเครื่องพัฒนาโปรแกรมที่อินเทลได้ออกแบบ

ในขณะนั้นอุปกรณ์เพอริเฟอรัลจำพวกเทป กระดาษเจาะรู และเทเลไทพ์ (teletype) ยังคงมีใช้อยู่ หากคุณไม่เคยใช้อุปกรณ์เหล่านี้คงไม่ทราบถึงเสียงดังที่น่ารำคาญ และความล่าช้าของการทำงานอุปกรณ์ดังกล่าวจึงไม่เป็นที่นิยมใช้กัน แต่อย่างไรก็ตามในช่วงเวลาของการพัฒนาCP/M แกรี่ยังคิดคำนึงถึงอุปกรณ์ดังกล่าว ในขณะเดียวกันนั้นเอง ดิสเก็ตต์ก็เข้ามามีบทบาทมากขึ้น และแกรี่ก็เริ่มพัฒนาโปรแกรมควบคุมการจัดการแผ่นจานแม่เหล็กไปด้วยจน CP/M สำเร็จออกมาเป็นรูปเป็นร่าง

ระหว่างการแข่งขันกันในบริษัทขายไมโครคอมพิวเตอร์นั้นเอง บริษัทอิมไซได้ทราบเรื่องก็มาติดต่อขอซื้อสิทธิบัตร CP/M เพื่อนำไปใช้กับเครื่องไมโครคอมพิวเตอร์ จุดเริ่มต้นของ CP/M จึงได้เริ่มขึ้นเช่นเดียวกับจรวดที่พุ่งขึ้นจากฐานอย่างสวยงาม

ขณะที่มีการซื้อขายสิทธิบัตรของ CP/M นั้น แกรี่ได้ลาออกจากบริษัทอินเทลไปแล้ว และใช้ชื่อบริษัทที่เขาตั้งเองว่า ดิจิตอลรีเสิร์ช เน้นการพัฒนาซอฟต์แวร์เป็นหลัก CP/M จึงเป็นชื่อทางการค้าของบริษัทดิจิตอลรีเสิร์ชไป คำว่า CP/M นั้นดิจิตอลรีเสิร์ชได้ถือเอามาจากตัวย่อของคำว่า Control Program for Microcomputer

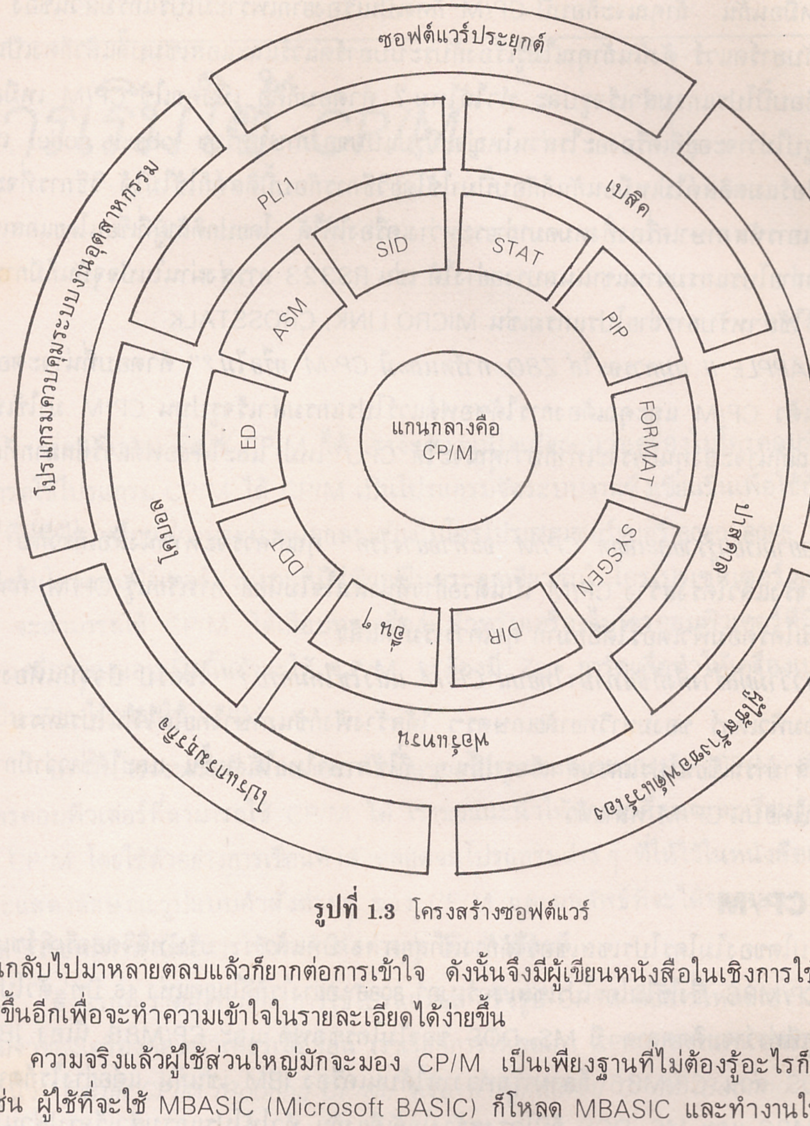
เพื่อให้ CP/M มีขีดความสามารถสูงขึ้นได้ ดิจิตอลรีเสิร์ชจึงได้เพิ่มโปรแกรมประกอบอีกหลายโปรแกรม เช่น แอสเซมเบลอร์ (ASM) เครื่องช่วยพิมพ์ (ED) เครื่องช่วยตรวจสอบโปรแกรม (DDT) โปรแกรมเหล่านี้ได้รับการเพิ่มเติมและสร้างเพิ่มเข้าไปเรื่อย ๆ

ทำไม CP/M แพร่หลาย

สืบเนื่องจาก CP/M เป็นโปรแกรม OS ที่ค่อนข้างจะแพร่หลายในหมู่เครื่องไมโครคอมพิวเตอร์ที่เขียนด้วยภาษาของไมโครโปรเซสเซอร์เบอร์ 8080 ดังนั้นจึงใช้ได้กับไมโครคอมพิวเตอร์ที่อยู่ในตระกูลเดียวกันคือ 8085 และ Z80 CP/M เปรียบเสมือนแกนกลางของระบบซอฟต์แวร์ เมื่อมี CP/M แล้วก็สามารถที่จะหาซอฟต์แวร์อื่น ๆ มาประกอบพิจารณาโครงสร้างการประกอบของระบบซอฟต์แวร์ที่มี CP/M เป็นแกนกลาง ดังรูปที่ 1.3

CP/M เสมือนกับเป็นแกนกลางที่มีอยู่ในระบบซอฟต์แวร์ของเครื่องไมโครคอมพิวเตอร์ CP/M เป็นระบบที่ต่อเข้ากับทางฮาร์ดแวร์ทำให้ระบบซอฟต์แวร์รอบนอกไม่เกี่ยวข้องกับฮาร์ดแวร์ วงรอบที่ติดกับ CP/M จึงเป็นวงรอบที่เรียกว่า *ยูทิลิตี้ (utility)* เช่น แอสเซมเบลอร์ DDT ฟอร์แมต ฯลฯ ในวงรอบนอกทำให้ผู้ใช้ใช้ง่ายขึ้น ซึ่งได้แก่แพ็คเกจต่าง ๆ หรือโปรแกรมประยุกต์ตลอดรวมไปถึงตัวแปรภาษาอื่น ๆ

ปัจจุบันเครื่องไมโครคอมพิวเตอร์มากกว่า 100 บริษัทที่ผลิตและสร้างขึ้นได้ซื้อลิขสิทธิ์ CP/M ไว้เป็น OS ของระบบแล้ว ดังนั้นตำราเกี่ยวกับ CP/M ในเชิงคู่มือเครื่องจึงมีเขียนขึ้นมากมาย โดยเนื้อแท้แล้วดิจิตอลรีเสิร์ชเองก็ได้ออกคู่มือ CP/M ขึ้นมา คู่มือดังกล่าวส่วนใหญ่ก็จะว่ากันในลักษณะผู้ใช้



รูปที่ 1.3 โครงสร้างซอฟต์แวร์

ซึ่งอ่านกลับไปมาหลายตลบแล้วก็ยากต่อการเข้าใจ ดังนั้นจึงมีผู้เขียนหนังสือในเชิงการใช้ CP/M แบบง่าย ๆ ขึ้นอีกเพื่อจะทำความเข้าใจในรายละเอียดได้ง่ายขึ้น

ความจริงแล้วผู้ใช้ส่วนใหญ่มักจะมอง CP/M เป็นเพียงฐานที่ไม่ต้องรู้อะไรก็ข้ามไปสู่ยอดได้เลย เช่น ผู้ใช้ที่จะใช้ MBASIC (Microsoft BASIC) ก็โหลด MBASIC และทำงานในโหมด BASIC ได้เลย หรือผู้ใช้จะฟอร์แทรนแทนที่จะเขียนด้วย ED ของ CP/M ผู้ใช้อาจจะข้ามไปใช้ WORDSTAR หรือโปรแกรมสำเร็จรูปเวิร์ดโปรเซสซิ่งได้

นอกจากนี้ผู้สร้างเครื่องไมโครส่วนใหญ่มักสร้างฟังก์ชันที่เรียกว่า *การเริ่มต้นอัตโนมัติ (auto start)* ซึ่งหมายถึงการเปิดเครื่องเพื่อโหลดโปรแกรมสำเร็จรูปได้เลย กล่าวคือ เมื่อเปิดเครื่องก็เป็นโปรแกรมสำเร็จรูปเลย

คำถามที่เกี่ยวข้องกับ CP/M

"ผมจะก๊อบปี้โปรแกรมจากเครื่องหนึ่งมายังอีกเครื่องหนึ่งได้ไหม ?" สมมติว่าคุณมีเครื่อง Osborn 1 ใช้ CP/M เพื่อนคุณมี APPLE ใช้ไมโครซอฟท์ Z80 การ์ด ทั้งสองเครื่องใช้ดิสค์ 5¼″ เหมือนกัน

แต่ฟอร์แมตไม่เหมือนกัน ถ้าคุณจะก๊อบปี้ CP/M ก็คงเป็นเรื่องยากเพราะมีโปรแกรมส่วนของ BIOS ที่ยังแตกต่างขึ้นกับฮาร์ดแวร์ ดังนั้นถ้าคุณไม่รู้เรื่องกับระบบฮาร์ดแวร์และแอสเซมบลี้แล้วก็คงเป็นเรื่องที่ยาก แต่ถ้าจะก๊อบปี้โปรแกรมสำเร็จรูปล่ะ ทำได้ไหม ? คำตอบก็คือ เมื่อคุณใช้ CP/M เหมือนกัน โปรแกรมสำเร็จรูปไม่ว่าจะอยู่ที่เครื่องอะไรส่วนใหญ่จะมีรูปแบบของภาษาเครื่อง (object code) เหมือนกัน ดังนั้นเมื่อฟอร์แมตดิสค์ไม่เหมือนกันก็ก๊อบปี้มาใช้โดยวิธีการก๊อบปี้ดิสค์ก็ใช้ไม่ได้ วิธีการที่จะใช้ได้ก็คือ หาทางนำเอารหัสภาษาเครื่องทั้งหมดมาถ่ายระหว่างเครื่องให้ได้ โดยปกติถ้าผู้ที่เขียนโดยแอสเซมบลี้ได้แล้วก็สามารถถ่ายโปรแกรมผ่านแชนเนลบางอย่างได้ เช่น RS323 การส่งผ่านในปัจจุบันก็มีการสร้างระบบซอฟต์แวร์ใช้สำหรับการถ่ายโปรแกรมเช่น MICRO LINK, CROSSTALK

"ผมมี APPLE II ผมควรจะใส่ Z80 การ์ดและมี CP/M หรือไม่?" คำตอบที่น่าจะตอบได้ก็คือ คุณสนใจในตัว CP/M และคุณต้องการได้ซอฟต์แวร์โปรแกรมสำเร็จรูปบน CP/M มาใช้เพียงไร ถ้าคุณต้องการคุณก็น่าจะลงทุนเพราะเท่ากับว่าคุณจะได้ CPU ใหม่ และได้ซอฟต์แวร์อีกมากทีเดียวที่อิงอยู่กับ CP/M

"ดิฉันอยากเรียนรู้รายละเอียด CP/M จะทำอย่างไรดี" คุณก็ควรจะหาหนังสือเกี่ยวกับ CP/M มาศึกษา ความจริงแล้วโครงสร้าง CP/M เป็นตัวอย่างที่น่าสนใจไม่น้อย การเรียนรู้ CP/M ทำให้คุณเข้าใจในระบบไมโครคอมพิวเตอร์ได้อีกมาก คุณควรเริ่มได้แล้ว

"ได้ข่าวว่ามีผู้สร้างฟังก์ชันภาษาไทยบน CP/M แล้วใช่ไหมครับ?" ใช่ครับ ปัจจุบันห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์ ของมหาวิทยาลัยเกษตรฯ ได้สร้างฟังก์ชันภาษาไทยใส่ไว้ในโปรแกรม CP/M เลยครับ ทำให้สามารถเขียนโปรแกรมสำเร็จรูปอื่น ๆ ที่ใช้ภาษาไทยได้ง่ายขึ้น และได้ข่าวว่ามีการผลิตซอฟต์แวร์ภาษาไทยบน CP/M หลายตัว

อนาคตของ CP/M

การเติบโตของไมโครโปรเซสเซอร์ได้ก้าวเข้าสู่ยุค 16 บิทแล้ว บริษัทดิจิตอลรีเสิร์ชเองก็ได้ทำการพัฒนา CP/M86 ซึ่งใช้ไมโครโปรเซสเซอร์เบอร์ 8086 อย่างไรก็ดีในยุคแห่ง 16 บิท ตัวโปรแกรมจัดระบบงานที่ดูทีท่าว่าจะติดตลาด มี MS–DOS ของไมโครซอฟต์ และ CP/M86 นี่เอง IBM ได้เลือก MS–DOS ส่วน CP/M86 ก็สามารถทำงานได้บนเครื่อง IBM เช่นกัน แต่อย่างไรก็ตามโครงสร้างของ CP/M86 และ MS–DOS ก็มีโครงสร้างใกล้เคียงกัน ทำให้โปรแกรมสำเร็จรูปส่วนใหญ่ใช้งานได้ทั้งสองระบบ

อนาคตของ CP/M86 หรือ MS–DOS ซึ่งใช้ชุดคำสั่งของ 8086 ซึ่งมีข้อดีประการหนึ่งคือ ระบบซอฟต์แวร์โปรแกรมสำเร็จรูปที่เขียนด้วย CP/M ของ 8080 สามารถเปลี่ยนแปลงเพื่อนำไปใช้บน 8086 ได้โดยไม่ยาก ทำให้ระบบซอฟต์แวร์ 8 บิท สามารถเข้าสู่ยุทธจักร 16 บิทได้เร็วกว่าปกติ

2

การเริ่มใช้ CP/M

ก่อนที่เราจะเริ่มใช้ CP/M ได้ เราจะต้องแน่ใจเสียก่อนว่าเครื่องไมโครคอมพิวเตอร์ที่เราใช้อยู่สามารถใช้โปรแกรม CP/M ได้ CP/M เป็นโปรแกรมจัดระบบงานซึ่งเขียนขึ้นเพื่อใช้กับไมโครคอมพิวเตอร์ที่มีซีพียู หรือหน่วยประมวลผลกลางเป็นไมโครโปรเซสเซอร์เบอร์ 8080 8085 หรือ Z80 ดังนั้นเครื่องไมโครคอมพิวเตอร์ส่วนมากที่มีซีพียูอยู่ในตระกูลเดียวกับไมโครโปรเซสเซอร์ 8080 8085 หรือ Z80 จะสามารถใช้ CP/M ได้เกือบทุกเครื่อง สำหรับเครื่องไมโครคอมพิวเตอร์ที่มีซีพียูเป็นอย่างอื่น เช่น เครื่องของแอปเปิ้ลนั้นถ้าจะใช้ CP/M จะต้องมี Z80 การ์ดเพื่อทำให้เครื่องเปลี่ยนซีพียูจากเดิมไปเป็น Z80 ในขณะใช้ CP/M

บทนี้ไปเราจะเริ่มต้นศึกษาการทำงานของโปรแกรมจัดระบบงาน CP/M สำหรับผู้ที่มีเครื่องไมโครคอมพิวเตอร์ที่สามารถใช้ CP/M ได้ เราขอแนะนำให้ทำการศึกษาและเรียนรู้การใช้คำสั่งต่าง ๆ ของ CP/M โดยใช้ตัวอย่างการเขียนคำสั่งตลอดจนโปรแกรมต่าง ๆ ที่ให้ไว้ในหนังสือเล่มนี้เป็นแนวทาง เราจะแสดงลักษณะรูปแบบคำสั่งต่าง ๆ ของ CP/M และผลลัพธ์ที่จะได้จากจอภาพเพื่อให้ผู้ศึกษาได้มีโอกาสเรียนรู้คำสั่งและการใช้งานคำสั่งต่าง ๆ ได้อย่างมีผลดี

สิ่งแรกที่เราต้องทำก่อนการใช้โปรแกรมสำเร็จรูป CP/M คือการโหลดเอา CP/M ลงในหน่วยความจำของเครื่องคอมพิวเตอร์เสียก่อน เราเรียกการโหลดเอา CP/M ลงในหน่วยความจำของคอมพิวเตอร์นี้ว่า *การบูท CP/M (Booting CP/M)* หรือ *การบูทแสตรป CP/M (Booting-strapping CP/M)*

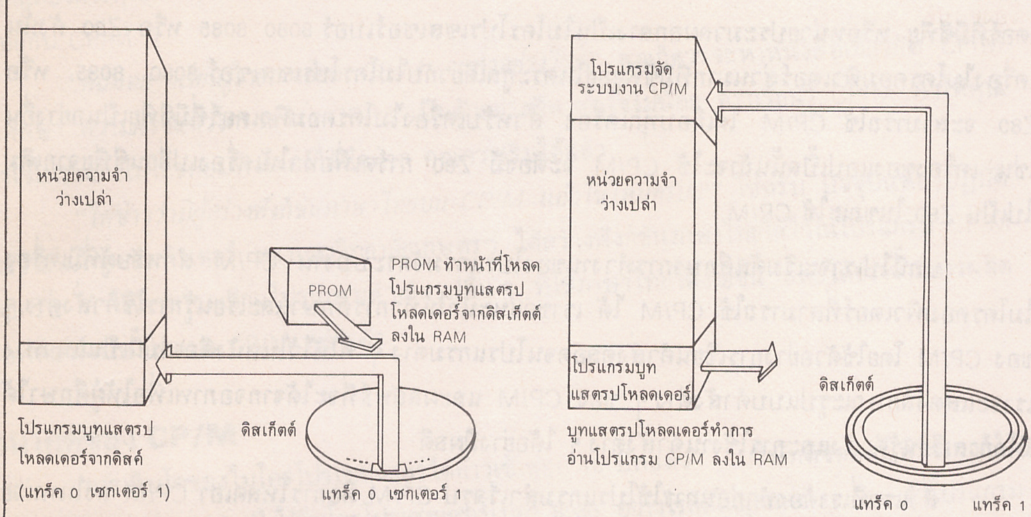
สำหรับบางคน เรื่องการบูท CP/M หรือการบูทโปรแกรมจัดระบบงานอื่น ๆ คงเป็นเรื่องที่ไม่เคยรู้มาก่อน ดังนั้นก่อนที่เราจะกล่าวถึงการใช้งาน CP/M เราลองมาศึกษาถึงการบูท CP/M กันก่อน

การบูท CP/M

ดังได้กล่าวมาแล้วว่าโปรแกรมจัดระบบงาน คือ โปรแกรมคอมพิวเตอร์ซึ่งเขียนขึ้นเพื่อทำให้เครื่องคอมพิวเตอร์สามารถอ่าน เขียน ข้อมูลและโปรแกรมอื่น ๆ จากหน่วยความจำภายนอก เช่น ฟลอปปี้ดิสค์ แต่เนื่องจาก CP/M เป็นโปรแกรมซึ่งเก็บอยู่ในฟลอปปี้ดิสค์หรือในดิสเก็ตต์ ดังนั้นก่อนที่เราจะสามารถใช้โปรแกรม CP/M ได้ เราจะต้องสามารถโหลดเอา CP/M ลงในหน่วยความจำเสียก่อน เรา

เรียกโปรแกรมซึ่งทำหน้าที่นี้ว่า บูทแสตรปโหลดเดอร์ (Bootstrap loader) ซึ่งโปรแกรมนี้โดยทั่วไปจะเก็บอยู่ในแทร็ค 0 เซกเตอร์ 1 ของดิสเก็ตต์

เครื่องไมโครคอมพิวเตอร์โดยทั่วไปจะมีโปรแกรมอยู่ใน ROM หรือ PROM เพื่อทำหน้าที่อ่านโปรแกรมจากส่วนหนึ่งส่วนใดในดิสเก็ตต์เข้าไปไว้ในหน่วยความจำเพื่อเริ่มจัดระบบการทำงานของเครื่อง ในขณะที่เราเปิดเครื่องหรือกดคีย์ RESET โดยทั่วไปโปรแกรมใน ROM หรือ PROM จะอ่านโปรแกรมบูทแสตรปโหลดเดอร์จากแทร็ค 0 เซกเตอร์ 1 ไปไว้ในหน่วยความจำที่มีแอดเดรสต่ำ ๆ จากนั้นโปรแกรมบูทแสตรปโหลดเดอร์จึงจะทำการโหลดเอาโปรแกรมจัดระบบงาน CP/M เข้าไว้ในหน่วยความจำ ซึ่งโดยทั่วไปจะไว้ในหน่วยความจำที่มีแอดเดรสสูง ภาพแสดงการโหลดโปรแกรมบูทแสตรปโหลดเดอร์และ CP/M ลงในหน่วยความจำแสดงอยู่ในรูปที่ 2 1



(ก) การโหลดโปรแกรมบูทแสตรปโหลดเดอร์จากแทร็ค 0 เซกเตอร์ 1 ลงใน RAM

(ข) การโหลด CP/M ลงใน RAM

รูปที่ 2.1 ภาพแสดงการโหลด CP/M ลงในหน่วยความจำ RAM

เมื่อเครื่องไมโครคอมพิวเตอร์ทำการบูท CP/M เสร็จแล้ว จะแสดงผลบนจอภาพคือ

```
XXK CP/M        VER y.y

A>
```

ซึ่งหมายความว่า CP/M จะทำการสำรองหน่วยความจำ XX กิโลไบท์แรกไว้สำหรับการใช้งานของ CP/M ค่าของ XX จะมีค่าระหว่าง 16 ถึง 64 พึงระลึกไว้ว่าขนาดของหน่วยความจำ RAM ในเครื่องคอมพิวเตอร์ที่เราใช้งานอยู่อาจมากกว่าค่า XX ที่ปรากฏ ดังนั้นเราอาจพบเสมอว่ามีการใช้ CP/M ขนาด 32 K

หรือ 48 K กับเครื่องไมโครซึ่งมีหน่วยความจำ RAM ขนาด 64 K y.y คือตัวเลขแสดงเวอร์ชัน (Version) ของโปรแกรม CP/M ที่กำลังใช้งานอยู่ เช่น เวอร์ชัน 2.2 หรือ 3.0 เป็นต้น สำหรับคำสั่งต่าง ๆ ในหนังสือเล่มนี้เป็นของ CP/M 2.2 ซึ่งจัดว่าเป็นเวอร์ชันที่มีการใช้งานกันอย่างกว้างขวางที่สุดในไมโครคอมพิวเตอร์ระดับ 8 บิท อย่างไรก็ตามสำหรับ CP/M 3.0 หรือ CP/M เวอร์ชันอื่น ๆ นั้น โดยหลักการทำงานและคำสั่งพื้นฐานต่าง ๆ จะเหมือนกับ CP/M 2.2 ทุกประการ แต่อาจมีรายละเอียดปลีกย่อยของการทำงานบางอย่างตลอดจนคำสั่งบางคำสั่งเท่านั้นที่แตกต่างออกไปจาก CP/M 2.2

เราเรียกกระบวนการบูท CP/M นี้ว่า โคลด์บูท (Cold Boot) ซึ่งเปรียบเสมือนกับการเริ่มติดเครื่องรถยนต์ในตอนเช้า โดยลักษณะที่เครื่องคอมพิวเตอร์เริ่มเปิดใช้งาน ส่วนลักษณะการบูทอีกแบบหนึ่งคือวอร์มบูท (Warm Boot) ซึ่งเป็นการบูท CP/M เมื่อเราทำการรันโปรแกรมหรือยูทิลิตี้โปรแกรม (Utility program) อยู่และต้องการกลับเข้าไปใน CP/M และ CCP การทำโคลด์บูทนั้นเครื่องคอมพิวเตอร์จะทำการโหลดโปรแกรม CP/M ทั้งหมดจากดิสเก็ตต์ในหน่วยความจำ ส่วนการทำวอร์มบูทนั้นเครื่องคอมพิวเตอร์จะทำการอ่านบางส่วนของ CP/M เท่านั้นลงสู่หน่วยความจำโดยถือว่าส่วนอื่น ๆ ของ CP/M มีอยู่แล้วในหน่วยความจำ

การจัดเนื้อที่ในหน่วยความจำ

ดังได้กล่าวมาแล้วว่าเครื่องคอมพิวเตอร์ทั่วไปประกอบด้วยองค์ประกอบที่สำคัญ 3 ส่วน คือ หน่วยสมองหรือซีพียู หน่วยความจำ และหน่วยอินพุต/เอาต์พุต สำหรับเครื่องคอมพิวเตอร์ที่ใช้โปรแกรมจัดระบบงาน CP/M นั้น จำนวนหน่วยความจำสูงสุดที่ซีพียูสามารถเรียกใช้งานได้คือ 65, 536 ไบท์ หรือ 64 กิโลไบท์ ซึ่งเรามักเรียกย่อ ๆ กันว่า 64 K (1 K = 1024 ไบท์) หน่วยความจำของคอมพิวเตอร์ทุกไบท์จะมีเลขแสดงตำแหน่งซึ่งเรียกว่า แอดเดรส เหมือนกับเลขที่บ้านที่เราอาศัยอยู่ แอดเดรสของหน่วยความจำจะมีค่าตั้งแต่แอดเดรสที่ 0 จนถึงแอดเดรสที่ 65, 535 ด้านแอดเดรสของหน่วยความจำแต่ละไบท์นี่เองที่ทำให้ซีพียูสามารถออกคำสั่งเพื่อทำการอ่าน เขียน หรือเคลื่อนย้ายข้อมูลระหว่างหน่วยความจำแต่ละหน่วย

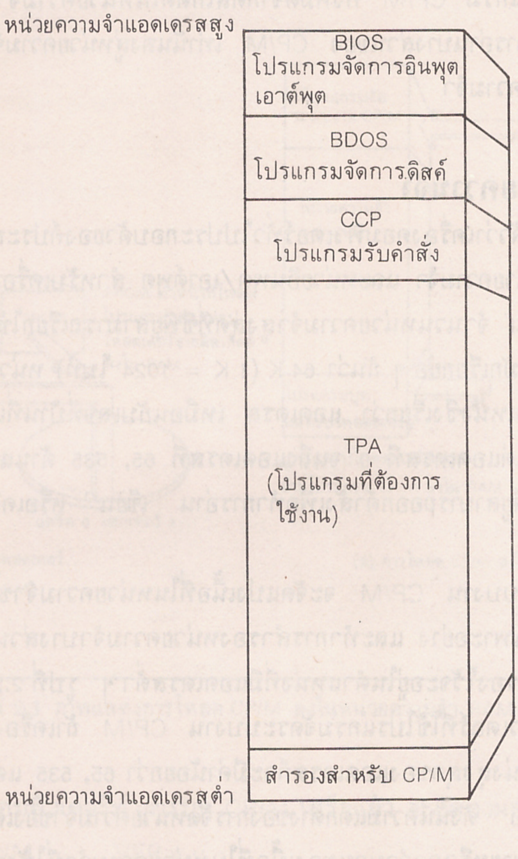
โปรแกรมจัดระบบงาน CP/M จะจัดแบ่งเนื้อที่ในหน่วยความจำของเครื่องคอมพิวเตอร์ออกเป็นส่วน ๆ เพื่อใช้งานเฉพาะอย่าง และทำการสำรองหน่วยความจำบางส่วนไว้เพื่อใช้งาน หน่วยความจำที่ CP/M ทำการสำรองไว้จะอยู่ในตำแหน่งที่มีแอดเดรสต่ำ ๆ รูปที่ 2.2 แสดงถึงการจัดเนื้อที่ในหน่วยความจำของคอมพิวเตอร์ที่ใช้โปรแกรมจัดระบบงาน CP/M ถ้าเครื่องคอมพิวเตอร์มีหน่วยความจำน้อยกว่า 64 K ตำแหน่งสูงสุดของแอดเดรสก็จะมีค่าน้อยกว่า 65, 535 แต่เนื่องจากตัวโปรแกรมของ CP/M มีขนาดคงที่เสมอ ดังนั้นความแตกต่างของการจัดหน่วยความจำของเครื่องที่มีขนาดหน่วยความจำมากกับน้อยจึงอยู่ที่การขยายหรือลดจำนวนของเนื้อที่ในหน่วยความจำที่ใช้งานของโปรแกรมเมอร์หรือผู้เขียนโปรแกรม คือส่วนของ TPA หรือทรานเซียนต์ โปรแกรมแอเรีย (Transient Program Area) แต่อย่างไรก็ตามในบางครั้งการจัดให้โปรแกรมซึ่งเขียนโดยโปรแกรมเมอร์ รันในหน่วยความจำนอกบริเวณ TPA อาจทำให้การทำงานของเครื่องคอมพิวเตอร์มีประสิทธิภาพมากขึ้น ข้อดีของ CP/M ประการสำคัญคือ ความสามารถในการรันโปรแกรมภายใต้หน่วยความจำชนิดต่าง ๆ และอุปกรณ์อินพุต/เอาต์พุต

ได้หลายชนิด ดังนั้นในคอมพิวเตอร์บางเครื่องอาจจัดให้ CP/M ใช้เนื้อที่ในหน่วยความจำ 42 K แรก โดยรวมกับส่วนที่เหลืออีก 24 K ไว้สำหรับการรันโปรแกรมโดยเฉพาะ การจัดเนื้อที่ในหน่วยความจำลักษณะนี้แสดงอยู่ในรูปที่ 2.3

ลักษณะความคล่องตัวของการจัดเนื้อที่ในหน่วยความจำด้วย CP/M นี้จัดการโดยโปรแกรมยูทิลิตี้ที่มีชื่อว่า MOVCPM โดยโปรแกรม MOVCPM จะทำหน้าที่จัดการโหลดโปรแกรมย่อยส่วนต่าง ๆ ของ CP/M คือ BDOS BIOS และ CCP ลงในหน่วยความจำตำแหน่งต่าง ๆ ตามแต่ขนาดหน่วยความจำของเครื่องคอมพิวเตอร์ที่ใช้อยู่ รายละเอียดการทำงานของ MOVCPM จะได้กล่าวต่อไปในบทถัดไป

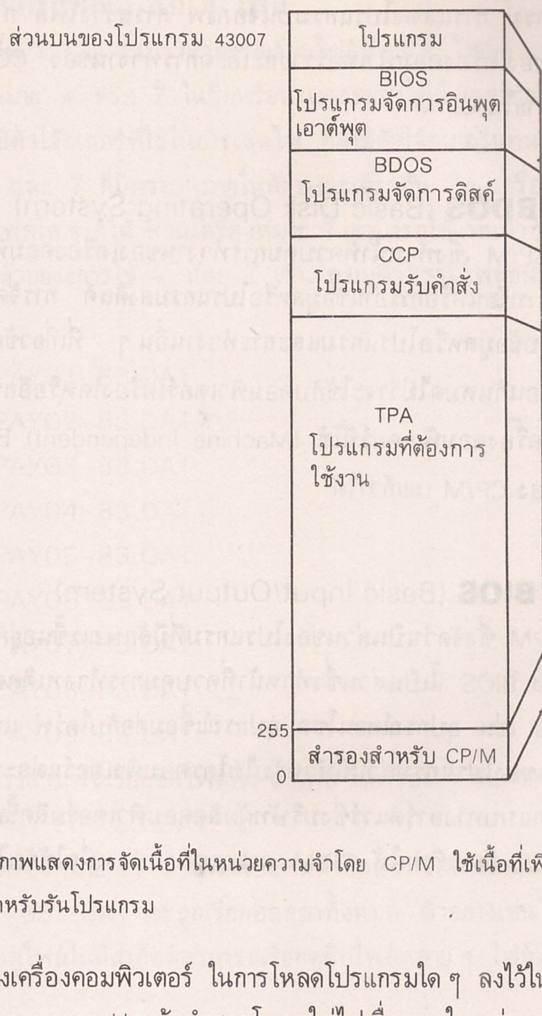
ทรานเชียนท์โปรแกรมแอเรีย TPA (Transient Program Area)

TPA คือเนื้อที่ในหน่วยความจำนับจากตำแหน่งแอดเดรสที่ 256 หรือ 0100H ในเลขฐาน 16



รูปที่ 2.2 ภาพแสดงการจัดเนื้อที่ในหน่วยความจำโดย CP/M ใช้เนื้อที่ทั้งหมด 64 K

จนถึงแอดเดรสก่อนแอดเดรสแรกของ BDOS TPA เป็นบริเวณในหน่วยความจำซึ่งผู้เขียนโปรแกรมสามารถเขียนโปรแกรมเก็บไว้ หรือเป็นส่วนซึ่งใช้ในการเก็บโปรแกรมซึ่งโหลดจากดิสเก็ตต์เข้ามาในหน่วย



รูปที่ 2.3 ภาพแสดงการจัดเนื้อที่ในหน่วยความจำโดย CP/M ใช้เนื้อที่เพียง 42 K และกันที่ 24 K เหนือบริเวณของ BIOS ไว้สำหรับรันโปรแกรม

ความจำของเครื่องคอมพิวเตอร์ ในการโหลดโปรแกรมใด ๆ ลงไว้ใน TPA คอมพิวเตอร์จะเริ่มต้นตำแหน่งแรกที่แอดเดรส 0100H แล้วทำการโหลดใส่ไปเรื่อย ๆ ในหน่วยความจำที่มีแอดเดรสมากขึ้นเรื่อย ๆ จนกว่าจะหมดโปรแกรม หรือจนกว่าเนื้อที่ในส่วน TPA จะไม่สังเกตว่าหน่วยความจำในส่วนของ CCP นั้นถือว่าเป็นส่วนหนึ่งของ TPA ด้วย ดังนั้นถ้าโปรแกรมที่ทำการโหลดลงใน TPA มีขนาดใหญ่มากเกินกว่าบริเวณปกติของ TPA ส่วนของโปรแกรมที่เหลือก็จะถูกโหลดลงใน CCP ด้วย เมื่อทำการรันโปรแกรมเสร็จแล้วและต้องการกลับไปใช้ส่วนของ CCP อีก เราก็ต้องทำการวอร์มบูทโดยกดคีย์ CTRL และ C พร้อม ๆ กัน เครื่องคอมพิวเตอร์ก็จะทำการโหลด CCP จากดิสเก็ตต์ลงไปไว้ในหน่วยความจำของเครื่อง

ขนาดเนื้อที่หน่วยความจำของ TPA จะมีขนาดไม่แน่นอนขึ้นอยู่กับขนาดของหน่วยความจำของเครื่องคอมพิวเตอร์ที่ใช้ แต่ CCP BDOS และ BIOS จะใช้ขนาดของหน่วยความจำเท่ากันเสมอ

คอนโซลคอมมานต์โปรเซสเซอร์ : CCP (Console Command Processor)

CCP เป็นส่วนหนึ่งของ CP/M ซึ่งทำหน้าที่ควบคุมการติดต่อระหว่างคนใช้กับเครื่องคอมพิวเตอร์เมื่อเราเรียกใช้คำสั่งหรือคอมมานด์ของ CP/M CCP ทำหน้าที่ควบคุมการโหลดโปรแกรมจากดิส

เก็ตต์ลงใน TPA และทำการรันโปรแกรม การแสดงโปรแกรมบนจอภาพ การสร้างไฟล์ การลบไฟล์ และการดูแลการทำงานในส่วนต่าง ๆ ของเครื่องคอมพิวเตอร์รายละเอียดการทำงานของ CCP จะได้กล่าวต่อไปเมื่อเรากล่าวถึงการใช้คำสั่งภายใน CP/M

เบสิคดิสค์โอเปอเรติงซิสเต็ม : BDOS (Basic Disk Operating System)

BDOS เป็นส่วนหนึ่งของ CP/M ซึ่งทำหน้าที่ควบคุมการทำงานของเครื่องคอมพิวเตอร์ในส่วนซึ่งเกี่ยวข้องกับการใช้ดิสค์ เช่น การเขียนหรือการอ่านข้อมูลหรือโปรแกรมลงดิสค์ การจัดสารหน่วยความจำในแผ่นดิสเก็ตต์เพื่อใช้ในการเก็บข้อมูลหรือโปรแกรมและการทำงานอื่น ๆ ที่เกี่ยวข้องกับดิสค์ โปรแกรมส่วนของ BDOS นี้จะเหมือนกันหมดไม่ว่าจะใช้กับคอมพิวเตอร์เครื่องใดหรืออีกนัยหนึ่งคือ BDOS เป็นโปรแกรมส่วนที่ไม่ขึ้นกับเครื่องคอมพิวเตอร์ที่ใช้ (Machine Independent) BDOS จัดว่าเป็นส่วนสำคัญหรือจัดว่าเป็นหัวใจของ CP/M เลยก็ว่าได้

เบสิคอินพุต/เอาต์พุตซิสเต็ม : BIOS (Basic Input/Output System)

BIOS เป็นส่วนหนึ่งของ CP/M ซึ่งจัดว่าเป็นส่วนของโปรแกรมที่มีลักษณะขึ้นอยู่กับเครื่องที่ใช้ (Machine Dependent) ส่วนของ BIOS นี้เป็นส่วนซึ่งทำหน้าที่ควบคุมการทำงานติดต่อระหว่างซีพียูกับอุปกรณ์อินพุต/เอาต์พุตทั้งหลาย เช่น อุปกรณ์คอนโซล อุปกรณ์เชื่อมต่อกับไดร์ฟ และอุปกรณ์เชื่อมต่อกับเครื่องพิมพ์ เป็นต้น ส่วนของโปรแกรมส่วนนี้สำหรับไมโครคอมพิวเตอร์แต่ละบริษัทอาจไม่เหมือนกัน แล้วแต่ลักษณะการออกแบบทางฮาร์ดแวร์ซึ่งบริษัทผู้ผลิตคอมพิวเตอร์ผลิตขึ้น การทำงานร่วมกันระหว่าง BDOS และ BIOS นี่เองที่ทำให้ CP/M สามารถดัดแปลงเพื่อใช้กับไมโครคอมพิวเตอร์ต่าง ๆ ได้มากมายหลายยี่ห้อ

ไฟล์ใน CP/M

สาเหตุหนึ่งซึ่งทำให้ CP/M ได้รับความนิยมใช้งานอย่างกว้างขวางก็เพราะความสามารถในการจัดการไฟล์ต่าง ๆ ได้เป็นอย่างดี *ไฟล์* คือ ที่รวมของข้อมูล ข้อความ หรือโปรแกรมต่าง ๆ ซึ่งเก็บอยู่ในฟลอปปี้ดิสค์หรือดิสเก็ตต์ไฟล์ใน CP/M นั้นสามารถตั้งชื่อด้วยจำนวนตัวอักษรสูงสุดไม่เกิน 8 อักขระ และมีประเภทของไฟล์หรือนามสกุลของไฟล์อีก 3 ตัวอักษร ตัวอย่างของชื่อไฟล์ใน CP/M เช่น

STAT.COM	GENDEDG.DAT	BOOK.TXT
PROGRAM.BAS	PROGRAM.PRN	BOOK.BAK

ชื่อของไฟล์อาจประกอบด้วยตัวอักขระพิเศษอื่น ๆ ที่ไม่ใช่เครื่องหมายต่อไปนี้คือ ? * . , = และ : ได้ ตัวอย่างของชื่อไฟล์ซึ่งห้ามเขียนใน CP/M เช่น

PAY12:79.DAT	ใช้อักขระพิเศษ : ไม่ได้
INVENTORY.DAT	ชื่อไฟล์ยาวเกิน 8 ตัวอักษร

การเรียกไฟล์ทีละหลาย ๆ ไฟล์

ใน CP/M นั้นเราสามารถเรียกไฟล์หลาย ๆ ไฟล์ด้วยการเขียนชื่อไฟล์เพียงชื่อเดียวได้โดยการใส่เครื่องหมาย * หรือ ? ในชื่อหรือนามสกุลของไฟล์ เราเรียกวิธีการนี้ว่าการใช้ไวด์การ์ด (Wild Card) หรือการใช้ตัวโจ๊กเกอร์ที่ใช้ในการเล่นไพ่ คือใช้ตัวโจ๊กเกอร์แทนตัวอื่น ๆ ได้ ใน CP/M การใช้เครื่องหมาย * และ ? ก็มีความหมายในลักษณะเดียวกัน คือ เครื่องหมาย * สามารถใช้แทนชื่อหรือนามสกุลของไฟล์ใด ๆ ก็ได้ ส่วนเครื่องหมาย ? สามารถใช้แทนตัวอักษรใด ๆ ก็ได้

ตัวอย่างการใช้ * และ ? เช่น สมมติว่าเรามีไฟล์ชนิด .DAT อยู 9 ไฟล์ในดิสเก็ตต์ โดยมีชื่อเป็น

PAY01-83.DAT
PAY02-83.DAT
PAY03-83.DAT
PAY04-83.DAT
PAY05-83.DAT
PAY06-83.DAT
PAY07-83.DAT
PAY08-83.DAT
PAY09-83.DAT

เราสามารถเรียกชื่อไฟล์ทั้ง 9 ไฟล์ ได้พร้อม ๆ กันด้วยชื่อไฟล์ว่า

PAY ?? - 83.DAT

ซึ่งในที่นี้เครื่องหมาย ?? สามารถใช้แทนตัวอักษรหรือตัวเลขใด ๆ ก็ได้ ดังนั้นชื่อไฟล์ใด ๆ ที่มีชื่อเป็น PAY---- -83. DAT จะถูกเรียกออกมาทั้งหมด ตัวอย่างเช่น การใช้คำสั่ง DIR (Divectony) ซึ่งเป็นการขอดูไฟล์ในดิสเก็ตต์สามารถเรียกดูชื่อไฟล์หลาย ๆ ไฟล์ได้ด้วยการใช้เครื่องหมาย ? ดังตัวอย่างเช่น

```
> A DIR PAY ?? - 83.DATA
PAY01-83.DAT
PAY02-83.DAT
PAY03-83.DAT
PAY04-83.DAT
PAY05-83.DAT
PAY06-83.DAT
PAY07-83.DAT
PAY08-83.DAT
PAY09-83.DAT
```

สำหรับเครื่องหมาย * มีความหมายเหมือนกับชุดของ ? หลาย ๆ ตัว ตั๋วอย่างของชื่อไฟล์ต่อไปนี้มีความหมายเหมือนกันคือ

? ? ? ? ? ? ? ? .BAS หรือ *.BAS

? ? ? ? ? ? ? ? ? . ? ? ? หรือ *.*

จากตัวอย่างข้างต้นชื่อไฟล์ชื่อแรกหมายถึงไฟล์ทุกไฟล์ที่เป็นชนิด .BAS หรือมีนามสกุล .BAS ส่วนชื่อไฟล์ที่สอง หมายถึงไฟล์ทุกไฟล์ คือ ไฟล์ทุกชื่อและทุกประเภท

ประเภทของไฟล์

ใน CP/M นั้นได้จัดแบ่งไฟล์ต่าง ๆ ออกเป็นประเภท (type) ตามแต่ลักษณะการใช้งานใน CP/M โปรแกรมยูทิลิตี้ที่เขียนอยู่บน CP/M หรือภาษาต่าง ๆ ที่ใช้ CP/M เป็นตัวจัดการระบบงานให้ เช่น ภาษา BASIC FORTRAN COBOL และอื่น ๆ ประเภทของไฟล์ถูกกำหนดโดยการตั้งนามสกุลด้วยอักษรย่อ 3 ตัว ประเภทของไฟล์ต่าง ๆ ใน CP/M ประกอบด้วย

.ASM	ไฟล์ภาษาแอสเซมบลี้ (Assembly)
.BAK	แบคอัฟไฟล์ (Backup file) ซึ่งสร้างโดยคำสั่ง ED
.BAS	ไฟล์โปรแกรมภาษาเบสิค (Basic)
.CRF	ไฟล์ชนิดครอสแอสเซมบลี้ เรเฟอร์เรนซ์ (Cross assembly reference)
.COB	ไฟล์โปรแกรมภาษาโคบอล (COBOL)
.DAT	ไฟล์ข้อมูล (DATA) ชนิด ASCII
.FOR	ไฟล์โปรแกรมภาษาฟอร์แทรน (FORTRAN)
.WT	ไฟล์ภาษาเครื่องของโปรแกรมภาษาเบสิค
.MAC	ไฟล์โปรแกรมภาษาแอสเซมบลี้ชนิดรีโลเคเทเบิล (Relocatable Assembler Code)
.LST	ไฟล์เก็บตัวโปรแกรม (listing) ของโปรแกรมภาษาเบสิค
.OVR	ไฟล์ชนิดโอเวอร์เลย์ (Overlay)
.PRN	ไฟล์เก็บโปรแกรมของภาษาฟอร์แทรน โคบอล แอสเซมบลี้และอื่น ๆ
.REL	ไฟล์ชนิดรีโลเคเทเบิลอ็อบเจ็กต์ (Relocatable Object) ของภาษาต่าง ๆ เช่น ฟอร์แทรน แอสเซมบลี้และอื่น ๆ
.SUB	คอมมานด์ไฟล์ SUBMIT
.SYM	ไฟล์เก็บรหัสของภาษาแอสเซมบลี้ (Assembler Symbol)
.XRF	ไฟล์เก็บรหัสแอสเซมเบลอร์ครอสเรเฟอร์เรนซ์ (Assembler Cross-Reference)
.$$$	ไฟล์ชั่วคราวที่สร้างโดยคำสั่ง ED หรือ PIP

การจัดไฟล์ในดิสเก็ตต์

ไฟล์ใน CP/M อาจมีจำนวนเรคอร์ดมากน้อยแตกต่างกันตั้งแต่ไฟล์ซึ่งไม่มีข้อมูลหรือโปรแกรมอยู่เลย จนถึงไฟล์ซึ่งอาจมีจำนวนข้อมูลเต็มแผ่นดิสเก็ตต์หน้าที่หลักประการหนึ่งของ BDOS ใน CP/M คือการจัดการเกี่ยวกับการจัดสรรเนื้อที่ในแผ่นดิสเก็ตต์เพื่อใช้เก็บไฟล์ต่าง ๆ เมื่อไฟล์มีขนาดใหญ่ขึ้น CP/M ก็จะทำการจัดสรรที่ในดิสเก็ตต์ให้แก่ไฟล์เพิ่มมากขึ้น และเมื่อไฟล์มีขนาดเล็กลง เช่น เมื่อมี

การลบเรคอร์ดบางส่วนในไฟล์ออกไป CP/M ก็จะทำการเรียกส่วนที่ว่างอยู่คืนจากไฟล์นั้นเพื่อเตรียมพร้อมที่จะจัดสรรให้แก่ไฟล์อื่นติดไปใหม่ซึ่งความสามารถในการเพิ่มขนาดของไฟล์และการนำบริเวณที่ไม่ใช้แล้วคืนจากไฟล์ทำให้ CP/M เป็นโปรแกรมจัดระบบงานซึ่งสามารถทำงานได้อย่างมีประสิทธิภาพเหนือกว่าโปรแกรมจัดระบบงานอื่น ๆ

กลไกในการจัดสรรเนื้อที่ให้แก่ไฟล์ใน CP/M มีลักษณะคล้ายคลึงกับวิธีการจัดเก็บเอกสารต่าง ๆ ซึ่งนักธุรกิจทำกัน แผ่นดิสเก็ตต์แต่ละแผ่นอาจถือเสมือนตู้เอกสารใบใหญ่โดยมีแต่ละไฟล์เปรียบเสมือนลิ้นชักเก็บเอกสารต่าง ๆ เมื่อเราต้องการเรียกใช้เอกสารชนิดใดจากไฟล์เราก็ทำการเปิดตู้เอกสารในลิ้นชักที่เก็บเอกสารนั้นไว้ออกมาใช้

CP/M จัดเก็บข้อมูลต่าง ๆ โดยแบ่งออกเป็นเรคอร์ดโดยมีความยาวเรคอร์ดละ 128 ไบท์ ในแต่ละไฟล์สามารถเก็บข้อมูลไว้ได้กี่เรคอร์ดก็ได้ตราบเท่าที่ยังมีเนื้อที่ว่างเหลืออยู่ในแผ่นดิสเก็ตต์ CP/M สามารถเรียกเรคอร์ดในไฟล์เพื่อนำไปไว้ในหน่วยความจำเพื่อทำการตรวจสอบแก้ไข และนำมากลับไปเก็บไว้ในตำแหน่งเดิมในดิสเก็ตต์ได้ ตัวอย่างเช่น เราอาจต้องการตรวจสอบเรคอร์ดที่ 20 ในไฟล์ชื่อ PAYROL.DAT CP/M จะทำการนำข้อมูลเรคอร์ดที่ 20 เขียนลงในหน่วยความจำ แสดงให้เราเห็นบนจอภาพ หลังจากที่เราทำการตรวจสอบแก้ไขเรียบร้อยแล้ว CP/M ก็จะนำข้อมูลกลับไปเก็บไว้ในเรคอร์ดที่ 20 ของไฟล์ได้

CP/M เก็บตำแหน่งของเรคอร์ดต่าง ๆ ในไฟล์ไว้ในส่วนที่เรียกว่า *ไดเรกทอรี (Directory)* ไดเรกทอรีของไฟล์แต่ละไฟล์จะเก็บรายละเอียดต่าง ๆ ของไฟล์ เช่น ชื่อของไฟล์ ชนิดของไฟล์ จำนวนเรคอร์ดในไฟล์ ตำแหน่งของเรคอร์ดต่าง ๆ ว่าอยู่ในแทร็คใด เซกเตอร์ใด เป็นต้น การสร้างไฟล์และการลบไฟล์บนแผ่นดิสเก็ตต์สามารถทำได้โดยการเขียนหรือลบไดเรกทอรีของไฟล์นั่นเอง

จนถึงขณะนี้เราได้เรียนรู้ลักษณะโครงสร้างของ CP/M วิธีการบูท CP/M ตลอดจนรายละเอียดการจัดเนื้อที่ในหน่วยความจำและการสร้างไฟล์ต่าง ๆ ในแผ่นดิสเก็ตต์ ส่วนคำสั่งต่าง ๆ ของ CP/M จะได้กล่าวถึงในบทต่อ ๆ ไป

3

คำสั่งเบื้องต้นของ CP/M

ในบทก่อนหน้านี้เราได้ทราบแล้วว่าใน CP/M ประกอบด้วยส่วนต่าง ๆ 4 ส่วนคือ TPA CCP BDOS และ BIOS ในจำนวนองค์ประกอบของ CP/M ทั้ง 4 นี้ CCP เป็นส่วนซึ่งผู้ใช้ติดต่อด้วยมากที่สุด ดังนั้นก่อนที่เราจะเริ่มต้นศึกษาการทำงานของคำสั่งต่าง ๆ ใน CP/M เราจะอธิบายถึงรายละเอียดของ CCP ก่อน

คำสั่งใน CP/M สามารถแบ่งได้เป็นสองประเภทคือ *เรสิเดนต์คอมมานด์ (Resident Command)* หรือคำสั่งภายในของ CP/M และ *ทรานเชียนต์คอมมานด์ (Transient Command)* หรือคำสั่งภายนอก CP/M เรสิเดนต์คอมมานด์เป็นคำสั่งซึ่งอยู่ภายในส่วน CCP และเราสามารถเรียกใช้ได้ทันที ส่วนทรานเชียนต์คอมมานด์คือ คำสั่งซึ่งเป็นโปรแกรมเก็บอยู่ในแผ่นดิสเก็ตต์และไม่ได้เป็นส่วนของ CP/M โดยตรง แต่สามารถเรียกใช้คำสั่งเหล่านี้ได้โดยการโหลดโปรแกรมของทรานเชียนต์คอมมานด์ลงในส่วนของ TPA แล้วทำการรันโปรแกรมของทรานเชียนต์คอมมานด์ เราเรียกชื่อทรานเชียนต์คอมมานด์นี้อีกชื่อหนึ่งว่าโปรแกรมยูทิลิตี้ของระบบ

สัญญาณเตรียมพร้อม

มีเรสิเดนต์คอมมานด์ซึ่งเราสามารถเรียกใช้งานได้หลายคำสั่งด้วยกัน คำสั่งต่าง ๆ เหล่านี้จะถูกเอ็กซิคิว (Execute) ภายใน CCP เมื่อเครื่องได้ทำงานตามคำสั่งเสร็จแล้วส่วนของ CCP จะแสดงเครื่องหมายเตรียมพร้อมเพื่อรับคำสั่งต่อไป เราเรียกเครื่องหมายนี้ว่า*เครื่องหมายพรอมต์ (Prompt Sign)* เครื่องหมายเตรียมพร้อมนี้ประกอบด้วยตัวอักษร A B C หรือ D แล้วตามด้วยเครื่องหมายมากกว่า > และเคอร์เซอร์ (Cursor) ซึ่งเตรียมพร้อมที่จะรับตัวอักษรต่อไป ดังแสดงคือ

```
A > ■
│ │ └─→ เคอร์เซอร์
│ └───→ เครื่องหมายเตรียมพร้อมของ CP/M
└─────→ อักษรแสดงว่าขณะนี้เครื่องคอมพิวเตอร์กำลังติดต่ออยู่กับดิสค์ไดร์ฟ A
```

อักษร A B C หรือ D แทนดิสค์ไดร์ฟที่เครื่องคอมพิวเตอร์กำลังทำงานหรือติดต่อด้วยเมื่อเริ่มทำการบูท CP/M นั้นเราต้องใส่ดิสเก็ตต์ของ CP/M ลงในดิสค์ไดร์ฟ A ดังนั้นเมื่อเริ่มใช้ CP/M หลังจากทำการบูท CP/M เสร็จแล้ว ส่วนของ CCP จะสั่งให้เครื่องคอมพิวเตอร์แสดง A< ออกมา

ในกรณีที่เราต้องการเปลี่ยนไปใช้ดิสค์ไดร์ฟอื่น เช่น ไดร์ฟ B เราสามารถทำได้โดยการคีย์อักษร B แล้วตามด้วย : ดังแสดงต่อไปนี้

ดิสค์ไดร์ฟเมื่อเริ่มต้น

A>B: ← ดิสค์ไดร์ฟที่ต้องการติดต่อด้วย

B>

แสดงว่าขณะนี้เครื่องคอมพิวเตอร์ต่อกับไดร์ฟ B แล้ว

ในทำนองเดียวกันกับการกลับไปทำงานกับดิสค์ไดร์ฟ A เราก็สามารถคีย์ดังนี้คือ

B>A:

A>

ในกรณีที่เราพยายามจะติดต่อกับดิสค์ไดร์ฟที่ไม่ได้ต่อกับเครื่องคอมพิวเตอร์ ตัวอย่างเช่น ถ้าเราพยายามเรียกใช้ดิสค์ไดร์ฟ D โดยที่เครื่องคอมพิวเตอร์ต่ออยู่กับดิสค์ไดร์ฟเพียง 2 ไดร์ฟ เครื่องก็จะหยุดการทำงานเพราะไม่สามารถติดต่อกับไดร์ฟ D ได้ ซึ่งเราจะต้องทำการโคลด์บูท CP/M ใหม่ โดยการคีย์ปุ่ม RESET หรือปิดสวิตช์ของซีพียูแล้วเปิดเครื่องใหม่

เรสิเดนต์คอมมานด์

CP/M มีคำสั่งภายในหรือเรสิเดนต์คอมมานด์อยู่ 5 คำสั่งประกอบด้วย

DIR	คำสั่งเพื่อขอดูไดเรกทอรีของไฟล์
ERA	คำสั่งเพื่อลบไฟล์ออกจากแผ่นดิสเก็ตต์
REN	คำสั่งเพื่อเปลี่ยนชื่อไฟล์
TYPE	คำสั่งเพื่อแสดงผลข้อมูลหรือโปรแกรมในไฟล์
SAVE	คำสั่งเพื่อเก็บข้อมูลหรือโปรแกรมจากหน่วยความจำลงในแผ่นดิสเก็ตต์

จากนี้ไปเราจะได้อธิบายการทำงานของคำสั่งภายในแต่ละคำสั่งอย่างละเอียด

คำสั่ง DIR

DIR เป็นคำสั่งซึ่งเขียนย่อมาจากคำว่า DIRECTORY เป็นคำสั่งซึ่งทำการแสดงไดเรกทอรีของไฟล์ในดิสเก็ตต์ออกมา คำสั่งนี้เป็นคำสั่งที่เรามักจะใช้งานกันบ่อยที่สุด คำสั่ง DIR สามารถทำให้เราขอดูไดเรกทอรีของไฟล์ในดิสค์ไดร์ฟใดก็ได้ โดยจะดูทุกไฟล์หรือเลือกดูเฉพาะไฟล์ก็ได้ พิจารณาการใช้คำสั่ง DIR ดังต่อไปนี้

```
A>
A>DIR
A: STAT     COM : FORMAT   COM : COPY     COM : CONFIGIO BAS
A: PIP      COM : ED       COM : ASM      COM : DDT      COM
A: LOAD     COM : RW13     COM : APDOS    COM : SUBMIT   COM
A: XSUB     COM : DOWNLOAD COM : DUMP     ASM : DUMP     COM
A: CPM56    COM : BOOT     COM : MBASIC   COM : GBASIC   COM
A: TSC      COM : AUTORUN  COM : MFT      COM
A>
```

คำสั่ง DIR ข้างต้นทำหน้าที่แสดงไดเรกทอรีของไฟล์ทั้งหมดในดิสค์ไดร์ฟที่กำลังติดต่ออยู่กับเครื่องคอมพิวเตอร์ในขณะนั้นนั่นคือ ไดร์ฟ A

```
A>DIR B:*.*
B: TBASE3    COM : MRUN      COM : TBASE1    COM : KUFX80    COM
B: TEST      CMD : KU        DBF : PIP       COM : DBASEMSC  OVR
B: DBASEJOI  OVR : DBASEAPP  OVR : DBASETTL  OVR : DBASEUPD  OVR
B: DBASESRT  OVR : DBASEBRO  OVR : DBASEMAI  OVR : DBASERPG  OVR
B: DBASEMOD  OVR : DBASEMSG  COM : DBASE     COM : TBASE2    COM
B: NUK       BAK : NUK       CMD
```

คำสั่ง DIR ข้างต้นเป็นการแสดงไดเรกทอรีของไฟล์ทั้งหมดในดิสค์ไดร์ฟ B

```
A>
A>DIR PIP.COM
A: PIP       COM
A>
```

คำสั่ง DIR ข้างต้นเป็นการตรวจสอบว่ามีไฟล์ชื่อ PIP COM อยู่ในดิสค์ไดร์ฟ A หรือไม่ ถ้ามีเครื่องจะแสดงชื่อไฟล์ออกมา แต่ถ้าไม่มีเครื่องจะไม่แสดงชื่อไฟล์เลย ตัวอย่างเช่น

```
A>DIR XX.COM
NO FILE
```

ในการตรวจสอบไฟล์ใดไฟล์หนึ่งว่ามีอยู่ในดิสค์ไดร์ฟอื่นหรือไม่สามารถทำได้โดยการใส่ชื่อของดิสค์ไดร์ฟลงหน้าชื่อไฟล์ดังตัวอย่าง เช่น

```
A>
A>DIR B:PIP.COM
B: PIP       COM
A>
```

ในการขอดูไดเรกทอรีของไฟล์ประเภทใดประเภทหนึ่งหรือชื่อใดชื่อหนึ่งเพียงอย่างเดียวสามารถทำได้โดยการใช้เครื่องหมาย * กับชื่อของไฟล์ดังตัวอย่างที่แสดงต่อไปนี้

```
A>DIR *.COM
A: STAT      COM : FORMAT    COM : COPY      COM : PIP       COM
A: ED        COM : ASM       COM : DDT       COM : LOAD      COM
A: RW13      COM : APDOS     COM : SUBMIT    COM : XSUB      COM
A: DOWNLOAD  COM : DUMP      COM : CPM56     COM : BOOT      COM
A: MBASIC    COM : GBASIC    COM : TSC       COM : AUTORUN   COM
A: MFT       COM
A>
```

```
A>DIR B:DBASE*.*
B: DBASEMSC  OVR : DBASEJOI  OVR : DBASEAPP  OVR : DBASETTL  OVR
B: DBASEUPD  OVR : DBASESRT  OVR : DBASEBRO  OVR : DBASEMAI  OVR
B: DBASERPG  OVR : DBASEMOD  OVR : DBASEMSG  COM : DBASE     COM
A>
```

คำสั่ง TYPE

TYPE เป็นคำสั่งซึ่งใช้ในการสั่งให้คอมพิวเตอร์แสดงผลข้อมูลหรือโปรแกรมที่อยู่ในไฟล์ออกทางอุปกรณ์แสดงผล เช่น บนจอภาพซีอาร์ที หรือทางเครื่องพิมพ์ คำสั่ง TYPE เหมาะสำหรับการตรวจสอบหรือดูสิ่งที่มีอยู่ในไฟล์ประเภทเท็กซ์ไฟล์ (Text File) ซึ่งเป็นไฟล์ที่เก็บข้อมูลด้วยรหัส ASCII สำหรับไฟล์ประเภทอื่น ๆ ซึ่งเป็นไฟล์ที่เก็บข้อมูลในลักษณะของรหัสควบคุมหรือรหัสที่เครื่องใช้ในการทำงานนั้น ถึงแม้จะแสดงผลออกมาได้ด้วยคำสั่ง TYPE แต่เราก็ไม่สามารถเข้าใจสิ่งที่แสดงผลออกมา เพราะสิ่งเหล่านั้นจะอยู่ในลักษณะของรหัสซึ่งเครื่องเข้าใจได้เท่านั้น ไฟล์ซึ่งเมื่อเรา TYPE ออกมาแล้วเราสามารถอ่านได้เข้าใจได้แก่ ไฟล์ประเภทเท็กซ์ไฟล์ ไฟล์ชั่วคราว และไฟล์โปรแกรมภาษาต่างๆ ตัวอย่างประเภทของไฟล์ซึ่งเราสามารถ TYPE ออกมาได้แก่ .BAS .FOR .ASM .MAC .PRN .LST และอื่น ๆ ตัวอย่างต่อไปนี้แสดงให้เห็นถึงการใช้คำสั่ง TYPE

```
A>
A>TYPE B:KUFX80.BAS
5 S=161
10 LPRINT CHR$(27);":";CHR$(0);CHR$(0);CHR$(0);
20 LPRINT CHR$(27);"%";CHR$(1);CHR$(0);
25 FOR T = 1  TO 7
30 LPRINT CHR$(27);"&";CHR$(0);
35 IF T = 7 THEN LPRINT CHR$(S);CHR$(S+3);:GOTO 50
40 LPRINT CHR$(S);CHR$(S+14);
50 LPRINT CHR$(139);
60 FOR N = 1 TO 15
70 FOR I=1 TO 12
80 READ R
90 IF T=7 AND I=12 AND N=4 THEN LPRINT CHR$(27);"1";:END
92 IF I=12 AND N<15 THEN LPRINT CHR$(139);:GOTO 120
94 IF I=12 AND N=15 THEN GOTO 130
100 LPRINT CHR$(R);
110 NEXT I
120 NEXT N
130 LPRINT:S=S+15
140 NEXT T
2000            REM CHR$(161),A1          1
2010 DATA 16,48,79,64,64,64
2020 DATA 64,64,63,0,0,0
2030            REM CHR$(162),A2          2
2040 DATA 32,64,126,1,1,1
2050 DATA 126,0,0,0,0,0
2060            REM CHR$(163),A3          3
2070 DATA 63,64,68,72,72,64
2080 DATA 64,64,63,0,0,0
2090            REM CHR$(164),A4          4
2100 DATA 66,37,94,100,4,2
2110 DATA 1,1,126,0,0,0
```

```
2120           REM CHR$(165),A5          5
2130 DATA 0,0,8,4,34,65
2140 DATA 126,0,0,0,0,0
2150           REM CHR$(166),A6          6
2160 DATA 32,64,64,80,80,72
2170 DATA 68,63,0,0,0,0
2180           REM CHR$(167),A7          7
2190 DATA 36,72,79,65,65,66
2200 DATA 68,63,6,0,0,0
2210           REM CHR$(168),A8          8
2220 DATA 32,64,95,33,1,33
2230 DATA 94,128,0,0,0,0
```

คำสั่ง TYPE ข้างต้นเป็นการสั่งให้คอมพิวเตอร์แสดงผลไฟล์ที่มีชื่อว่า KUFX80.BAS ทางอุปกรณ์แสดงผล ถ้าอุปกรณ์แสดงผลของเครื่องคอมพิวเตอร์เป็นจอภาพซีอาร์ที สิ่งต่าง ๆ ในไฟล์จะถูกนำออกแสดงทางจอภาพ ถ้าหากว่าสิ่งที่มีอยู่ในไฟล์มีจำนวนมากเกินกว่าที่จะแสดงได้ด้วยจอภาพคือมีความยาวเกินกว่า 24 บรรทัด จอภาพก็จะเลื่อนขึ้นไปเรื่อย ๆ จนทำให้เราไม่สามารถอ่านสิ่งต่าง ๆ ที่แสดงอยู่ได้ การสั่งให้จอภาพหยุดนิ่งอยู่กับที่ไม่เลื่อนขึ้นนั้นสามารถทำได้โดยการกดคีย์ CTRL และ S พร้อม ๆ กัน และเมื่อเราต้องการให้จอภาพแสดงผลส่วนที่เหลือออกมาก็กดคีย์อะไรก็ได้อีกครั้งหนึ่ง

ชื่อไฟล์ที่อยู่หลังคำสั่ง TYPE จะใช้ชื่อไฟล์ที่มีเครื่องหมาย * หรือ ? อยู่ด้วยไม่ได้ ดังนั้นตัวอย่างคำสั่ง TYPE ต่อไปนี้เป็นคำสั่งที่ผิด

```
A>^C
A>TYPE *.BAS
*.BAS?

A>TYPE *.*
*.*?

A>TYPE PROG.*
PROG.*?

A>TYPE ??????.BAS
??????.BAS?
```

อย่างไรก็ตามเราสามารถสั่งให้คอมพิวเตอร์ทำการแสดงผลของไฟล์หลาย ๆ ไฟล์ได้พร้อมกันโดยคีย์ชื่อไฟล์ต่อกันไป ดังตัวอย่างเช่น

A > TYPE PRO1.BAS , PRO2.BAS , BOOK.TXT DATA1.DAT

คำสั่ง ERA

ERA เป็นคำสั่งซึ่งย่อมาจากคำว่า ERASE ซึ่งแปลว่าการลบหรือล้างไฟล์คำสั่ง ERA จะทำให้ไฟล์ซึ่งระบุชื่อตามหลังคำสั่งนี้ถูกลบออกจากดิสเก็ตต์ตัวอย่าง เช่น

A > ERA PROGRAM1.BAS

คำสั่งข้างต้นจะทำให้ไฟล์ชื่อ PROGRAM1.BAS ถูกลบออกจากดิสเก็ตต์คำสั่งต่อไปนี้เป็นการลบไฟล์ชื่อ PRO2.BAS ซึ่งอยู่ในดิสค์ไดร์ฟ B : ออกจากดิสเก็ตต์ในไดร์ฟ B :

A > ERA B:PRO2.BAS

คำสั่ง ERA นั้นสามารถรับชื่อไฟล์ซึ่งใช้เครื่องหมาย * และ ? ได้ ตัวอย่างเช่น

A > ERA *.BAS

คำสั่งนี้จะทำให้ไฟล์ทุกไฟล์ที่เป็นประเภท .BAS ถูกลบออกจากดิสเก็ตต์ในไดร์ฟ A: ส่วนคำสั่ง

A > ERA B:PROGRAM.*

ทำให้ไฟล์ทุกประเภทที่มีชื่อว่า PROGRAM ถูกลบออกจากดิสเก็ตต์ที่อยู่ในดิสค์ไดร์ฟ B:

ในกรณีที่เราต้องการลบไฟล์ทุกไฟล์ออกจากแผ่นดิสเก็ตต์ เราสามารถสั่งด้วยคำสั่ง

A > ERA *.*

เมื่อคอมพิวเตอร์ได้รับคำสั่งนี้จะแสดงข้อความถามเพื่อความแน่ใจว่าเราต้องการลบไฟล์ทุกไฟล์ออกจริงหรือด้วยข้อความว่า ALL FILES (Y/N) ? ถ้าเราต้องการลบหมดทุกไฟล์ก็กดคีย์ Y แต่ถ้าไม่ต้องการลบทั้งหมดก็สามารถยกเลิกคำสั่งด้วยการกดคีย์ N เครื่องก็จะยกเลิกคำสั่ง ERA

เพื่อป้องกันการความผิดพลาดที่อาจเกิดขึ้นเนื่องจากการใช้คำสั่ง ERA ซึ่งอาจทำให้เราลบไฟล์บางไฟล์โดยไม่ตั้งใจ ก่อนการใช้คำสั่ง ERA ควรจะใช้คำสั่ง DIR เพื่อเรียกไดเรกทอรีของไฟล์ออกมาดูเสียก่อนที่จะทำการลบไฟล์ด้วยคำสั่ง ERA และก่อนการกดคีย์ RETURN ในคำสั่ง ERA ควรตรวจสอบให้แน่ใจเสียก่อนว่าเป็นไฟล์ที่เราต้องการลบออกจากดิสเก็ตต์จริง นอกจากนั้นในทางปฏิบัติแล้วควรหลีกเลี่ยงการใช้ชื่อไฟล์โดยมีเครื่องหมาย * โดยไม่จำเป็น เพราะการใช้ชื่อไฟล์ที่มีเครื่องหมาย * อาจทำให้ไฟล์ที่เราไม่ตั้งใจลบออกถูกลบออกจากแผ่นดิสเก็ตต์ได้

คำสั่ง **REN**

REN เป็นคำสั่งที่ย่อมาจากคำว่า RETURN ซึ่งมีความหมายว่าให้ทำการเปลี่ยนชื่อของไฟล์เสียใหม่ รูปแบบของคำสั่งคือ

A > REN ชื่อไฟล์ใหม่ = ชื่อไฟล์เก่า

ตัวอย่างเช่น

A > REN TEXT.BAK = TEXT.TXT

ชื่อไฟล์เก่า

ชื่อไฟล์ใหม่ที่ต้องการตั้งให้

คำสั่ง REN ข้างต้นจะทำการเปลี่ยนชื่อไฟล์ในดิสเก็ตต์ซึ่งแต่เดิมมีชื่อว่า TEXT.TXT ไปเป็นชื่อ TEXT.BAK การเปลี่ยนชื่อไฟล์นี้อาจทำได้ทั้งการเปลี่ยนชื่ออย่างเดียวแต่คงประเภทของไฟล์ไว้ หรือเปลี่ยนประเภทของไฟล์แต่คงชื่อเดิมไว้ (ดังตัวอย่างข้างต้น) หรือเปลี่ยนทั้งชื่อและประเภทของไฟล์ดังตัวอย่างต่าง ๆ ของการใช้คำสั่ง REN ดังต่อไปนี้

A > REN PRO1.BAS = PROGRAM. BAS

A > REN PRO2.BAS = PROGRAM2.BAK

A > REN PRO3.BAK = PRO3.TXT

ในกรณีที่เราต้องการเปลี่ยนชื่อไฟล์จากดิสค์ไดร์ฟหนึ่งแล้วนำไปไว้ในอีกดิสค์ไดร์ฟหนึ่ง เช่น

A > REN B:TEXT.BAK = TEXT.TXT

เป็นการเปลี่ยนชื่อไฟล์ที่ชื่อว่า TEXT.TXT ซึ่งอยู่ในดิสค์ไดร์ฟ A: ไปเป็น TEXT.BAK แล้วเก็บไว้ในดิสเก็ตต์ในดิสค์ไดร์ฟ B:

ถ้าหากชื่อของไฟล์ที่ต้องการเปลี่ยนชื่อไม่มีอยู่ในดิสเก็ตต์คอมพิวเตอร์ก็จะแสดงข้อความบอกเราด้วยข้อความว่า

NOT FOUND

และถ้าชื่อของไฟล์ที่ต้องการตั้งใหม่เกิดไปซ้ำกับชื่อที่มีอยู่แล้วในดิสเก็ตต์เครื่องก็จะบอกให้เรารู้ด้วยการแสดงข้อความว่า

FILE EXISTS

ซึ่งเราต้องเปลี่ยนชื่อไฟล์ที่ต้องการตั้งใหม่ไปเป็นอย่างอื่นที่ไม่ซ้ำไฟล์ที่มีอยู่ในดิสเก็ตต์หรืออาจจะเปลี่ยนชื่อไฟล์ที่มีชื่อซ้ำในดิสเก็ตต์ไปเป็นอย่างอื่นเสียก่อนจึงค่อยทำการเปลี่ยนชื่อไฟล์ที่ต้องการได้

เพื่อหลีกเลี่ยงความสับสนซึ่งอาจเกิดขึ้นได้ดังนั้นคำสั่ง REN จึงไม่อนุญาตให้ใช้ชื่อไฟล์ที่มีเครื่องหมาย * อยู่ด้วยในคำสั่ง

คำสั่ง SAVE

SAVE เป็นคำสั่งสุดท้ายใน CCP ซึ่งใช้งานเกี่ยวข้องกับการโปรแกรมภาษาแอสเซมบลีเป็นหลัก ดังนั้นเราจะกล่าวถึงคำสั่ง SAVE อย่างละเอียดอีกครั้งในบทที่เกี่ยวข้องกับเรื่อง DDT และ ASM ในบทนี้จะกล่าวแต่เพียงย่อ ๆ ถึงหน้าที่และการทำงานของคำสั่ง SAVE พอเป็นสังเขป

คำสั่ง SAVE จะทำการนำโปรแกรมหรือข้อมูลจำนวน n เพจ (page) (1 เพจ = 256 ไบท์) จากหน่วยความจำบริเวณ TPA ไปเก็บไว้ในไฟล์ของดิสเก็ตต์ ตัวอย่างคำสั่ง SAVE ต่อไปนี้ทำการเก็บข้อมูลจาก 5 เพจแรกของ TPA คือจากตำแหน่งที่ 100H ถึง 4FFH ลงในไฟล์ชื่อ

A > SAVE 5 PROGRAM.COM

ทรานเชียนต์คอมมานด์ (ยูทิลิตี้โปรแกรม)

คำสั่งใน CP/M ซึ่งใช้งานได้สะดวกที่สุดและทำงานได้รวดเร็วที่สุดคือชุดคำสั่งของเรสิเดนต์คอมมานด์ซึ่งได้กล่าวถึงแล้ว ทั้งนี้เนื่องจาก CP/M ไม่ต้องโหลดโปรแกรมเข้ามาเพื่อทำคำสั่งดังกล่าว โปรแกรมของคำสั่งดังกล่าวมีอยู่ภายใน CCP แล้วการทำงานจึงทำให้ได้รวดเร็ว แต่อย่างไรก็ตามโปรแกรมของคำสั่งดังกล่าวก็ทำให้คอมพิวเตอร์ต้องจัดสรรหน่วยความจำบางส่วนให้จึงทำให้เนื้อที่ใช้งานน้อยลง ด้วยเหตุนี้นี่เองจึงไม่สามารถบรรจุเรสิเดนต์คอมมานด์ได้หลายคำสั่งลงใน CCP ดังนั้นคำสั่งอื่น ๆ ซึ่งต้องการสร้างขึ้นเพื่อใช้ทำงานตามที่เราต้องการจึงต้องเขียนเป็นโปรแกรมแยกต่างหากออกจาก CP/M

และเก็บเป็นไฟล์ไว้ในดิสเก็ตต์ เมื่อไรที่ต้องการเรียกใช้คำสั่ง CP/M จะทำการโหลดโปรแกรมของคำสั่งเหล่านั้นเข้ามาใน TPA แล้วทำการรันโปรแกรมดังกล่าวเราเรียกคำสั่งใน CP/M ซึ่งเก็บไว้เป็นโปรแกรมในไฟล์นี้ว่า *ทรานเชียนต์คอมมานด์*

โปรแกรมของคำสั่งประเภททรานเชียนต์คอมมานด์บางครั้งเรียกว่ายูทิลิตี้โปรแกรม โปรแกรมเหล่านี้จะเป็นประเภท COM ไฟล์ ซึ่ง .COM ไฟล์มาตรฐานที่ใช้เป็นคำสั่งใน CP/M มีอยู่หลายคำสั่ง ตัวอย่างเช่น

STAT.COM	ED.COM	DDT.COM
ASM.COM	PIP.COM	LOAD.COM
DUMP.COM	SYSGEN.COM	MOVCPM.COM

นอกจากคำสั่งต่าง ๆ ข้างต้น บริษัทผู้เขียนซอฟต์แวร์และผู้ผลิตเครื่องคอมพิวเตอร์อาจสร้างทรานเชียนต์คอมมานด์ขึ้นใช้เองเพื่อช่วยทำให้การใช้งาน CP/M ทำได้อย่างสะดวกและมีฟังก์ชันให้เลือกใช้มากขึ้นได้ คำสั่งทรานเชียนต์คอมมานด์ต่าง ๆ จะได้กล่าวรายละเอียดในบทต่อ ๆ ไป

การเรียกใช้คำสั่งทรานเชียนต์คอมมานด์ทำได้ในลักษณะคล้ายคลึงกับการเรียกใช้คำสั่งเรสิเดนต์คอมมานด์ คือเรียกใช้แต่เพียงการคีย์ชื่อคอมมานด์โดยไม่ต้องคีย์ประเภทของไฟล์ .COM ตามลงไปด้วย ตัวอย่างเช่น

A > STAT

เป็นการเรียกใช้คำสั่ง STAT

ทำนองเดียวกันกับเรสิเดนต์คอมมานด์ การเรียกใช้ทรานเชียนต์คอมมานด์ส่วนใหญ่จะต้องใส่ข้อความชื่อไฟล์ หรือสิ่งที่ต้องการให้ทำตามคำสั่งลงไปด้วยท้ายคำสั่ง ซึ่งถ้าเราไม่ใส่รายละเอียดที่สำคัญให้แก่คำสั่ง คอมพิวเตอร์ก็จะถามคำถามต่าง ๆ หรือบอกให้เราใส่ข้อความตามลงไปด้วย ซึ่งรายละเอียดต่าง ๆ จะได้เห็นต่อไป

4

FORMAT SYSTEM และ MOVCPM

ในบทที่ 2 เราได้กล่าวถึงองค์ประกอบต่าง ๆ ใน CP/M เราได้เห็นแล้วว่าส่วนต่าง ๆ ของ CP/M คือ CCP BDOS และ BIOS ทำงานและติดต่อกันอย่างไร ในบทนี้เราจะมาดูกันว่าเราจะนำ CP/M ไปใช้กับไมโครคอมพิวเตอร์ได้อย่างไร และที่สำคัญคือเราจะปรับปรุง CP/M ให้ใช้กับไมโครคอมพิวเตอร์ที่มีซีพียูเป็น 8080 8085 หรือ Z80 ได้อย่างไร

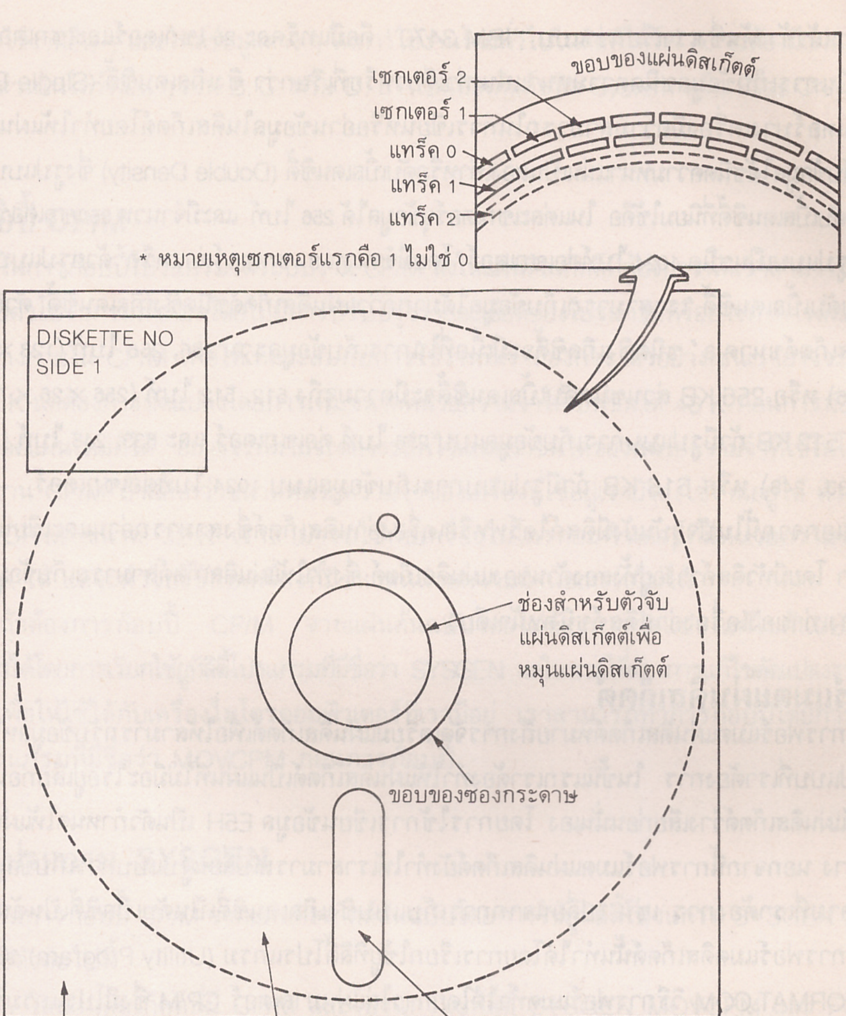
การเก็บข้อมูลในดิสเก็ตต์

สิ่งสำคัญประการแรกก่อนที่เราจะใช้ CP/M ได้ เราจะต้องทำให้แผ่นดิสเก็ตต์เปล่าสามารถเก็บข้อมูล โปรแกรม และรายละเอียดต่าง ๆ ให้ได้เสียก่อนและสิ่งที่เราไม่ควรละเลยก็คือ การทำการก๊อบปี้แผ่นดิสเก็ตต์ CP/M ที่มีอยู่เพื่อเอาไว้เป็นแผ่น CP/M สำรอง เผื่อว่าเราพลาดท่าเผลอทีทำให้ CP/M ที่ใช้งานอยู่เกิดถูกลบหรือทำลายไปโดยไม่ตั้งใจ โดยปกติสำหรับแผ่นดิสเก็ตต์ที่มีโปรแกรมสำคัญ ๆ อยู่ เราควรจะทำการก๊อบปี้เผื่อไว้เป็นแผ่นสำรองไม่น้อยกว่า 2 แผ่นเสมอ

สำหรับผู้ที่ยังไม่คุ้นเคยหรือยังไม่ทราบว่าดิสเก็ตต์นั้นเก็บข้อมูลต่าง ๆ ได้อย่างไร หัวข้อนี้คงเป็นประโยชน์ได้ไม่น้อย แต่สำหรับผู้ที่มีความเข้าใจในเรื่องเหล่านี้แล้วอาจข้ามหัวข้อ 3 หัวข้อต่อไปนี้และไปเริ่มต้นเรื่องการก๊อบปี้ CP/M

ข้อมูลต่าง ๆ ที่เก็บในแผ่นดิสเก็ตต์จะถูกเก็บเป็นกลุ่ม ๆ หรือเป็นบล๊อกโดยแต่ละบล๊อกอาจประกอบด้วย 128 หรือ 256 หรือ 1024 ไบท์ เราเรียกแต่ละบล๊อกว่า *เซกเตอร์ (Sector)* แต่ละเซกเตอร์จะมีแอดเดรสหรือตำแหน่งที่กำหนดไว้แน่นอนบนดิสเก็ตต์ การเก็บข้อมูลหรือการเรียกใช้ข้อมูลใด ๆ จากดิสเก็ตต์ก็ทำได้โดยการสั่งให้ตัวควบคุมดิสค์หรือที่เรียกว่าดิสค์คอนโทรลเลอร์ ทำการเขียนหรืออ่านข้อมูลจากเซกเตอร์ที่ต้องการ แต่ละเซกเตอร์ในดิสเก็ตต์จะถูกแบ่งตามแนวรัศมีของดิสเก็ตต์โดยมีจำนวนเซกเตอร์ที่แน่นอนจำนวนหนึ่ง นอกจากนี้แผ่นดิสเก็ตต์ยังแบ่งการเก็บข้อมูลออกเป็นร่องหรือแทร็ค (track) ตามแนวเส้นรอบวง คล้าย ๆ กับร่องของแผ่นเสียง แต่ละร่องเราเรียกว่า แทร็ค ภาพแสดงรายละเอียดการจัดเซกเตอร์และแทร็คของแผ่นดิสเก็ตต์แสดงอยู่ในรูปที่ 4.1

จำนวนเซกเตอร์และแทร็คของดิสเก็ตต์ขึ้นอยู่กับขนาดของแผ่น ตัวอย่างเช่น แผ่นดิสเก็ตต์ขนาดเส้นผ่านศูนย์กลาง 8″ โดยทั่วไปจะมี 26 เซกเตอร์และ 77 แทร็คต่อดิสเก็ตต์ 1 หน้า แต่ละเซกเตอร์ของ 1 แทร็ค มีหน่วยความจำ 128 ไบท์ ดังนั้นถ้าเราต้องการเขียนข้อมูลลงในแทร็คที่ 44 เซกเตอร์ที่ 23 ดิสค์คอน



รูปที่ 4.1 แสดงโครงสร้างของดิสเก็ตต์ ที่แบ่งเป็นแทร็ค/เซกเตอร์

โทรลเลอร์จะสั่งให้หัวดิสค์เคลื่อนที่ไปยังตำแหน่งแทร็คที่ 44 ขณะเดียวกันจะสั่งให้เครื่องจับแผ่นดิสเก็ตต์หมุนดิสเก็ตต์ไปจนตรงกับเซกเตอร์ที่ 23 หัวดิสค์จึงจะทำการเขียนข้อมูลลงในแผ่นดิสเก็ตต์ในตำแหน่งที่ต้องการ การอ่านข้อมูลจากตำแหน่งใด ๆ ในแผ่นดิสเก็ตต์ก็ทำได้เช่นเดียวกัน

การจัดเรียงแทร็คและเซกเตอร์แบบต่าง ๆ

ดังได้กล่าวมาแล้วในหัวข้อที่แล้วว่าการเก็บข้อมูลในแผ่นดิสเก็ตต์ของเครื่องไมโครคอมพิวเตอร์โดยทั่วไปจะจัดเก็บได้ 3 แบบ คือ 128 256 และ 1204 ไบท์ต่อเซกเตอร์ แบบที่นิยมใช้กันมากที่สุดคือแบบที่

ได้กล่าวมาแล้วข้างต้นซึ่งเราเรียกว่าแบบ ''IBM 3470'' คือมีแทร็คละ 26 เซกเตอร์และเซกเตอร์ละ 128 ไบท์ ซึ่งเป็นการเก็บข้อมูลชนิดความหนาแน่นเท่าเดียวหรือที่เรียกว่า ซิงเกิลเดนซิตี้ (Single Density) ในคอมพิวเตอร์บางเครื่องมีความสามารถในการเขียนหรืออ่านข้อมูลในดิสเก็ตต์โดยทำให้แผ่นดิสเก็ตต์สามารถเก็บข้อมูลได้ชนิดความหนาแน่นเป็นสองเท่าหรือดับเบิ้ลเดนซิตี้ (Double Density) ซึ่งรูปแบบการเก็บข้อมูลแบบดับเบิ้ลเดนซิตี้ที่นิยมใช้คือ ในแต่ละเซกเตอร์จุข้อมูลได้ 256 ไบท์ และมีจำนวน 26 เซกเตอร์ต่อแทร็ค หรืออาจมีรูปแบบเป็นชนิด 1024 ไบท์ต่อเซกเตอร์ โดยมีจำนวน 8 เซกเตอร์ต่อแทร็ค ด้วยรูปแบบการเก็บข้อมูลแบบดับเบิ้ลเดนซิตี้ เราสามารถเก็บข้อมูลได้มากกว่าแผ่นดิสเก็ตต์ชนิดซิงเกิลเดนซิตี้ ตัวอย่างเช่น สำหรับดิสเก็ตต์ขนาด 8″ ชนิดซิงเกิลซิตี้จะมีเนื้อที่ในการเก็บข้อมูลรวม 256, 256 ไบท์ (128 × 26 × 77 = 256, 256) หรือ 256 KB ส่วนชนิดดับเบิ้ลเดนซิตี้จะมีความจุถึง 512, 512 ไบท์ (256 × 26 × 77 = 512, 512) หรือ 512 KB ถ้ามีรูปแบบการเก็บข้อมูลแบบ 256 ไบท์ ต่อเซกเตอร์ และ 633, 248 ไบท์ (1024 × 8 × 77 = 633, 248) หรือ 616 KB ถ้ามีรูปแบบการเก็บข้อมูลแบบ 1024 ไบท์ต่อเซกเตอร์

นอกจากนี้ในปัจจุบันยังมีดิสค์ไดร์ฟหรือเครื่องอ่านดิสเก็ตต์ซึ่งสามารถอ่านและเขียนดิสเก็ตต์ได้สองหน้า โดยมีหัวดิสค์วางอยู่ทั้งสองด้านของแผ่นดิสเก็ตต์ ซึ่งทำให้แผ่นดิสเก็ตต์สามารถเก็บข้อมูลได้มากขึ้นเป็นสองเท่าของเครื่องอ่านดิสค์ชนิดหน้าเดียว

การฟอร์แมตแผ่นดิสเก็ตต์

การฟอร์แมตแผ่นดิสเก็ตต์หมายถึงการจัดเตรียมแผ่นดิสเก็ตต์เพื่อให้สามารถรับข้อมูลหรือเก็บข้อมูลตามรูปแบบที่เราต้องการ ในขั้นแรกเราต้องทำให้แผ่นดิสเก็ตต์เป็นแผ่นที่ไม่มีอะไรอยู่เสียก่อนหรือต้องการทำให้แผ่นดิสเก็ตต์ว่างเสียก่อนนั่นเอง โดยการใช้การเขียนข้อมูล E5H เป็นตัวกำหนดให้แผ่นดิสเก็ตต์เป็นแผ่นว่าง นอกจากนี้การฟอร์แมตแผ่นดิสเก็ตต์ยังทำให้เราสามารถเปลี่ยนรูปแบบการเก็บข้อมูลในแผ่นดิสเก็ตต์ตามที่เราต้องการ เช่น เปลี่ยนจากการเก็บแบบซิงเกิลเดนซิตี้เป็นดับเบิ้ลซิตี้เป็นต้น

การฟอร์แมตดิสเก็ตต์นั้นทำได้โดยการเรียกใช้ยูทิลิตี้โปรแกรม (Utility Program) ของ CP/M ที่ชื่อว่า FORMAT.COM วิธีการฟอร์แมตทำได้โดยการใส่แผ่นมาสเตอร์ CP/M ซึ่งมีโปรแกรมชื่อ FORMAT.COM อยู่ลงในดิสค์ไดร์ฟ A: และใส่แผ่นดิสเก็ตต์ที่ต้องการฟอร์แมตลงในดิสค์ไดร์ฟ B: แล้วพิมพ์คำสั่งว่า FORMAT ดังแสดง

A > FORMAT

จากนั้นโปรแกรม FORMAT จะถามคำถามต่าง ๆ เกี่ยวกับการฟอร์แมต เช่น ต้องการฟอร์แมตแบบซิงเกิลหรือดับเบิ้ลเดนซิตี ต้องการเก็บข้อมูลแบบ 128 256 หรือ 1025 ไบท์ต่อเซกเตอร์ เมื่อเราบอกสิ่งที่ต้องการแก่เครื่องคอมพิวเตอร์แล้ว เครื่องก็จะทำการฟอร์แมตแผ่นดิสเก็ตต์ตามที่เราต้องการแล้วออกจากส่วน CCP ของ CP/M

เมื่อเราได้ทำการฟอร์แมตแผ่นดิสเก็ตต์เป็นที่เรียบร้อยแล้ว เราก็สามารถใช้ดิสเก็ตต์แผ่นนี้เก็บข้อมูล โปรแกรมหรือสิ่งต่าง ๆ ได้ แต่เนื่องจากแผ่นดิสเก็ตต์นี้ยังไม่มีโปรแกรมจัดระบบงาน CP/M อยู่ในแผ่น ดังนั้นแผ่นนี้จะใช้งานได้แต่เฉพาะกับแผ่นอื่นที่มี CP/M อยู่ด้วยเท่านั้น เราเรียกดิสเก็ตต์ลักษณะเช่นนี้ว่า ''Files Only'' ดิสเก็ตต์ คือเก็บได้แต่ไฟล์ ส่วนแผ่นดิสเก็ตต์ที่มีตัวโปรแกรมจัดระบบงานอยู่ด้วยเราเรียกว่า ''System'' ดิสเก็ตต์ แผ่น System ดิสเก็ตต์นอกจากจะมีโปรแกรมจัดระบบงานอยู่ยังอาจประกอบด้วยโปร

แกรมการใช้งานอื่น ๆ และไฟล์ข้อมูลต่าง ๆ ในการใช้งานโดยทั่วไปในดิสค์ไดร์ฟ A: จะต้องมีแผ่น System ดิสเก็ตต์ ส่วนในไดร์ฟอื่น ๆ เช่น B: C: หรือ D: อาจเป็นดิสเก็ตต์ชนิด System หรือ Files Only ดิสเก็ตต์ก็ได้

การก๊อบปี้ CP/M

ในการก๊อบปี้โปรแกรมจัดระบบงาน CP/M ลงในแผ่นดิสเก็ตต์ที่ต้องการนั้นเราสามารถทำได้ทั้งการก๊อบปี้ตัวโปรแกรมทั้งหมดโดยไม่มีการปรับปรุงรายละเอียดของโปรแกรมหรืออาจทำการดัดแปลงบางสิ่งบางอย่างภายใน CP/M เพื่อให้เหมาะสมกับการใช้งานที่เราต้องการ ตัวอย่างเช่นเราอาจจะมี CP/M ขนาด 32 K แต่ต้องการดัดแปลงโดยการเพิ่มขนาดหน่วยความจำให้เป็นขนาด 48 K ก่อนการก๊อบปี้ลงในแผ่นดิสเก็ตต์แผ่นใหม่ก็ได้ อย่างไรก็ตามจะต้องระลึกไว้เสมอว่าขนาดของหน่วยความจำที่ใช้ในโปรแกรมจัดระบบงาน CP/M. ถ้าเล็กกว่าขนาดหน่วยความจำของเครื่องที่ใช้อยู่เรายังคงใช้งานอยู่ได้ ตัวอย่างเช่น ถ้าเรามี CP/M ขนาด 32 K เราสามารถใช้กับเครื่องไมโครคอมพิวเตอร์ที่มีหน่วยความจำ 48 K หรือ 64 K ได้ แต่จะนำไปใช้กับเครื่องที่มีขนาดหน่วยความจำน้อยกว่าไม่ได้

ถ้าต้องการก๊อบปี้ CP/M จากแผ่นต้นฉบับโดยไม่ต้องการดัดแปลงแก้ไขต้นฉบับเลยเราสามารถทำได้โดยการเรียกใช้ยูทิลิตี้โปรแกรมที่มีชื่อว่า SYSGEN แต่ในกรณีที่ต้องการแก้ไขดัดแปลงรายละเอียดบางอย่างเพื่อให้ใช้ได้กับเครื่องไมโครคอมพิวเตอร์ที่เรามีอยู่ เราสามารถทำการก๊อบปี้โดยการเรียกใช้ยูทิลิตี้โปรแกรมที่มีชื่อว่า MOVCPM ก่อนการก๊อบปี้

การใช้โปรแกรม SYSGEN

ในการก๊อบปี้ CP/M จากแผ่นดิสเก็ตต์ต้นฉบับโดยการใช้ยูทิลิตี้โปรแกรม SYSGEN มีขั้นตอนการก๊อบปี้ดังต่อไปนี้

1. ใส่แผ่นดิสเก็ตต์ที่มี CP/M และโปรแกรมยูทิลิตี้ SYSGEN MOVCPM และ FORMAT ลงในดิสค์ไดร์ฟ

2. ใส่แผ่นดิสเก็ตต์ที่ต้องการก๊อบปี้ CP/M ซึ่งอาจเป็นดิสเก็ตต์เปล่าหรือแผ่นที่เราไม่ต้องการใช้งานข้อมูลต่าง ๆ อีกแล้วลงในดิสค์ไดร์ฟ B:

3. เรียกใช้โปรแกรม FORMAT และ SYSGEN เพื่อทำการก๊อบปี้ CP/M จากแผ่นดิสเก็ตต์ในไดร์ฟ A: ลงไปไว้ในไดร์ฟ B: โดยคำสั่งดังต่อไปนี้

```
A>FORMAT B:
FORMATTING COMPLETE
A>SYSGEN
SOURCE DRIVE NAME : A
DESTINATION DRIVE NAME : B
DESTINATION ON B, THEN TYPE RETURN
DESTINATION DRIVE NAME:
```

A>

คำสั่ง FORMAT ในบรรทัดแรกเป็นการเรียกใช้ยูทิลิตี้โปรแกรม FORMAT เพื่อทำการฟอร์แมตแผ่นดิสเก็ตต์ในไดร์ฟ B: เสียก่อนเพื่อเตรียมพร้อมที่จะรับโปรแกรม CP/M ในกรณีที่ต้องการฟอร์แมตแผ่นดิสเก็ตต์ในไดร์ฟอื่นเราก็สามารถทำได้โดยการใส่ชื่อไดร์ฟตามคำสั่ง FORMAT ตัวอย่างเช่น

FORMAT A:

จะเป็นการฟอร์แมตแผ่นดิสเก็ตต์ที่อยู่ในไดร์ฟ A: นั่นเอง

เมื่อเครื่องทำการฟอร์แมตแผ่นดิสเก็ตต์เสร็จแล้วก็จะบอกให้เรารู้โดยการแสดงข้อความว่า

FORMATTING COMPLETE

หลังจากนั้นเราก็เรียกใช้ยูทิลิตี้โปรแกรม SYSGEN เพื่อทำการก๊อบปี้ CP/M จากแผ่นดิสเก็ตต์ในไดร์ฟ A: ไปยังไดร์ฟ B: โดยเครื่องจะถามเราก่อนว่าแผ่นต้นฉบับของ CP/M อยู่ในไดร์ฟไหนโดยข้อความว่า

SOURCE DRIVE NAME:

ถ้าแผ่นต้นฉบับเราอยู่ในไดร์ฟ A เราก็คีย์อักษร A ลงไป เครื่องจะถามเราต่อไปว่าต้องการก๊อบปี้ CP/M-ไปไว้ในแผ่นดิสเก็ตต์ในไดร์ฟใด ด้วยข้อความว่า

DESTINATION DRIVE NAME:

เมื่อเราต้องการก๊อบปี้ไปไว้ในไดร์ฟ B เราก็คีย์อักษร B ลงไป เครื่องจะบอกให้เราทราบอีกครั้งถึงไดร์ฟที่เราต้องการก๊อบปี้ไปไว้ด้วยข้อความ

DESTINATION ON B, THEN TYPE RETURN

เมื่อเรากด RETURN คีย์ เครื่องจะถามเราต่อไปอีกว่าต้องการก๊อบปี้ไปไว้ในไดร์ฟอื่นอีกหรือไม่ด้วยข้อความ

DESTINATION DRIVE NAME :

ถ้าเรามีแผ่นดิสเก็ตต์ในไดร์ฟอื่น ๆ อีก เช่น C: หรือ D: ที่ต้องการทำการก๊อบปี้เอา CP/M ไว้ เราก็ใส่ชื่อไดร์ฟลงไป เครื่องจะทำการก๊อบปี้ CP/M ไว้ในดิสเก็ตต์ได้หลาย ๆ แผ่น พร้อม ๆ กันด้วยคำสั่ง SYSGEN คำสั่งเดียว แต่ถ้าเราต้องการก๊อบปี้ลงใน B: เพียงไดร์ฟเดียว เมื่อเครื่องแสดงข้อความเพื่อให้เราใส่ชื่อไดร์ฟใหม่ที่ต้องการก๊อบปี้ ให้ทำการกดคีย์ RETURN โดยไม่ต้องใส่ชื่อไดร์ฟอื่นอีก เครื่องก็จะทำการก๊อบปี้ CP/M ลงในดิสเก็ตต์แผ่นที่ต้องการ

การใช้โปรแกรม MOVCPM

มักจะพบอยู่เสมอว่าเราจำเป็นต้องดัดแปลงแก้ไข CP/M ที่มีอยู่เพื่อให้เหมาะสมกับขนาดของหน่วยความจำของเครื่องในการนี้เราสามารถทำได้โดยการก๊อบปี้ CP/M พร้อมกับการดัดแปลงไปในตัวในการดัดแปลง CP/M ส่วนนี้แตกต่างจากการแก้ไขในส่วนของ BIOS ภายใน CP/M เพื่อให้เหมาะสมกับฮาร์ดแวร์ การใช้ MOVCPM เป็นเพียงเปลี่ยนแปลงขนาดของหน่วยความจำซึ่ง CP/M จะทำงานหรือ run โปรแกรมเท่านั้น แต่ไม่ได้เปลี่ยนแปลงตัวโปรแกรม CP/M เลย

โดยทั่วไป CP/M ต้นฉบับมักจะเป็นชนิด 16 K ถ้าเราต้องการขยายขนาดของหน่วยความจำให้มากขึ้นเพื่อให้สอดคล้องกับเครื่องไมโครที่มีอยู่ เราสามารถทำได้โดยใช้ยูทิลิตี้โปรแกรม MOVCPM

โปรแกรมนี้ทำให้เราสามารถกำหนดขนาดของหน่วยความจำเท่าใดก็ได้ตามต้องการ รูปแบบของการเรียกใช้โปรแกรมนี้คือ

A > MOVCPM n*

เมื่อ n เป็นจำนวนหน่วยความจำที่ต้องการมีหน่วยเป็น KB และเครื่องหมาย * เป็นตัวบอกให้เครื่องทำการจองที่ในหน่วยความจำเพื่อถ่ายเอา CP/M ไปไว้ในดิสเก็ตต์ด้วยโปรแกรม SYSGEN ถ้าเราไม่ใส่ตัวเลขของ n โปรแกรม MOVCPM จะทำการตรวจสอบขนาดหน่วยความจำของเครื่องคอมพิวเตอร์ที่กำลังใช้งานอยู่ โดยเริ่มต้นจากตำแหน่งที่ 0100H และนับจำนวนหน่วยความจำสูงสุดที่มีอยู่ แล้วทำการสร้าง CP/M ลงในแผ่นดิสเก็ตต์ที่ต้องการโดยมีขนาดหน่วยความจำตามที่นับได้

สมมติว่าเรามี CP/M ขนาด 16 K อยู่และต้องการก๊อบปี้ CP/M โดยขยายขนาดหน่วยความจำเป็น 32 K โดยเก็บไว้ในดิสเก็ตต์ในไดร์ฟ B: การก๊อบปี้ทำได้โดยมีขั้นตอนดังต่อไปนี้

1. ใส่แผ่นดิสเก็ตต์ CP/M ต้นฉบับไว้ในดิสค์ไดร์ฟ A:

2. ใส่แผ่นดิสเก็ตต์ที่ต้องการก๊อบปี้ CP/M ไว้ในดิสค์ไดร์ฟ B:

3. ทำการฟอร์แมตแผ่นดิสเก็ตต์ในไดร์ฟ B แล้วเรียกโปรแกรม MOVCPM เพื่อทำการแก้ไขขนาดหน่วยความจำของ CP/M แล้วจึงเรียกโปรแกรม SYSGEN เพื่อทำการก๊อบปี้ CP/M จากไดร์ฟ A: ไปยัง B: ชุดคำสั่งและข้อความต่าง ๆ ในการก๊อบปี้ CP/M โดยใช้ MOVCPM ด้วยแสดงอยู่ดังต่อไปนี้

```
A >FORMAT B:
FORMATTING COMPLETE
A > MOVCPM 32*
READY FOR "SYSGEN" OR
"SAVE 32 CPM32.COM"
A > SYSGEN
SOURCE DRIVE NAME : A
DESTINATION DRIVE NAME : B
DESTINATION ON B, THEN TYPE RETURN
DESTINATION DRIVE NAME
A >
```

ชุดคำสั่งข้างต้นทำให้เราได้ดิสเก็ตต์ที่มี CP/M ขนาดหน่วยความจำ 32 KB เครื่องหมายดอกจัน * เป็นการบอก MOVCPM ให้ออกจากระบบเพื่อที่ว่าโปรแกรม SYSGEN จะได้ทำหน้าที่ก๊อบปี้ CP/M-ลงในไดร์ฟ B:

ทดสอบ CP/M

เมื่อเราก๊อบปี้ได้แผ่นดิสเก็ตต์ที่มี CP/M อยู่ เราสามารถทดสอบว่าแผ่นดังกล่าวมี CP/M ซึ่งสามารถทำงานได้หรือไม่โดยการนำแผ่นใหม่ที่ได้ใส่ลงในไดร์ฟ A: แล้วกดคีย์ RETURN ถ้า CP/M ที่มีอยู่ใช้งานได้เครื่องคอมพิวเตอร์จะขึ้นสัญลักษณ์บนจอภาพคือ

A>

ซึ่งสัญลักษณ์นี้แสดงว่าเครื่องกำลังทำงานอยู่กับดิสค์ไดร์ฟ A ส่วนเครื่องหมาย > เป็นสัญลักษณ์ว่ากำลังอยู่ใน CP/M แต่ถ้าเรากดคีย์ RETURN แล้วเครื่องไม่แสดงสัญลักษณ์ดังกล่าวบนจอภาพก็แสดงว่า CP/M ที่ก๊อบปี้มายังใช้งานไม่ได้ เราต้องทำการก๊อบปี้ใหม่ และหลังจากการก๊อบปี้ใหม่แล้วยังไม่สามารถใช้งานได้อีกก็อาจเป็นไปได้ว่าแผ่นดิสเก็ตต์ดังกล่าวเสียและใช้งานไม่ได้

เมื่อทำการก๊อบปี้ CP/M เสร็จ เราก็จะได้แผ่นดิสเก็ตต์ที่มี CP/M อยู่ แต่ยังไม่มีโปรแกรมหรือข้อมูลใด ๆ อยู่เลย ในบทต่อไปเราจะได้แสดงถึงการใช้ยูทิลิตี้โปรแกรมที่มีชื่อว่า PIP ซึ่งทำให้เราสามารถก๊อบปี้โปรแกรมอื่น ๆ จากดิสเก็ตต์แผ่นหนึ่งไปไว้ในอีกแผ่นหนึ่งได้ตามที่ต้องการ

มีบริษัทผู้ผลิตเครื่องไมโครหลายแห่งซึ่งเขียนยูทิลิตี้โปรแกรมซึ่งสามารถก๊อบปี้แผ่นดิสเก็ตต์ทั้งแผ่นลงในอีกแผ่นหนึ่งที่ต้องการ เช่นจากไดร์ฟ A: ไปไว้ในไดร์ฟ B: โดยทั่วไปโปรแกรมนี้จะมีชื่อว่า DISKCOPY หรือชื่ออื่นที่มีความหมายคล้ายกับชื่อนี้ โปรแกรมนี้ทำหน้าที่ก๊อบปี้ดิสเก็ตต์ทั้งแผ่นซึ่งแท้ที่จริงแล้วคือการเรียกใช้ยูทิลิตี้โปรแกรม SYSGEN และ PIP ติดต่อกันนั่นเอง รายละเอียดการใช้โปรแกรม PIP จะได้กล่าวในบทต่อไป

5

STAT และ PIP

ในบรรดายูทิลิตี้โปรแกรมทั้งหลายของ CP/M STAT และ PIP จัดว่าเป็นโปรแกรมที่มีการใช้งานกันมากที่สุด โปรแกรมทั้งสองนี้เป็นโปรแกรมชนิดคอมมานด์ไฟล์ คือมีนามสกุลของไฟล์เป็น .COM ดังนั้นการคำสั่งในการเรียกใช้คือ STAT และ PIP จึงจัดว่าเป็นประเภททรานเชียนต์คอมมานด์ คือทุกครั้งที่มีการเรียกใช้โปรแกรมทั้งสองนี้เครื่องคอมพิวเตอร์จะต้องทำการโหลดโปรแกรมลงในบริเวณของทรานเชียนต์โปรแกรม (Transient Program Area) ในหน่วยความจำเสียก่อนจึงจะสามารถทำการสั่งให้โปรแกรมทำงานได้

โปรแกรม STAT ทำหน้าที่ในการบันทึกสถิติต่าง ๆ ในแผ่นดิสเก็ตต์หรือเปรียบเสมือนเป็นแม่บ้านที่คอยดูแลความเรียบร้อยของการใช้งานในดิสเก็ตต์ ส่วนโปรแกรม PIP ทำหน้าที่ในการก๊อบปี้โปรแกรมข้อมูล หรือไฟล์ต่าง ๆ จากดิสเก็ตต์แผ่นหนึ่งไปไว้ยังอีกแผ่นหนึ่ง โดยทั่วไปแล้วโปรแกรมจัดระบบงานทุกโปรแกรมจะต้องมีโปรแกรมซึ่งทำหน้าที่ทั้งสองนี้อยู่ด้วยเสมอ แต่สำหรับโปรแกรมจัดระบบงานอื่นที่ไม่ใช่ CP/M อาจเรียกชื่อโปรแกรมแตกต่างไปจาก STAT และ PIP ทั้งนี้เนื่องจากการเคลื่อนย้ายโปรแกรมข้อมูล หรือไฟล์ต่าง ๆ ระหว่างดิสเก็ตต์ตลอดจนการบันทึกรายละเอียดของไฟล์ต่าง ๆ ขนาดของไฟล์และจำนวนของหน่วยความจำในดิสเก็ตต์ที่มีเหลืออยู่นับว่าเป็นสิ่งสำคัญซึ่งเรามักต้องใช้งานอยู่เสมอ

STAT และ PIP เป็นโปรแกรมที่เราต้องเรียกใช้อยู่เสมอเมื่อต้องการก๊อบปี้โปรแกรมหรือแผ่นดิสเก็ตต์ไว้เพื่อเก็บไว้เป็นแผ่นสำรอง ดังนั้นเราจึงควรให้ความสนใจกับการใช้โปรแกรมทั้งสองนี้ให้มาก

ความรู้ทั่วไปเกี่ยวกับ STAT

STAT เป็นโปรแกรมที่มีชื่อย่อมาจากคำว่า STATUS ซึ่งมีความหมายว่าการบอกสถานภาพของอะไรสักอย่าง ในที่นี้หมายถึง การบอกสถานภาพของแผ่นดิสเก็ตต์ STAT เป็นคำสั่งประเภททรานเชียนต์คอมมานด์ซึ่งเราสามารถเรียกใช้ได้ใน CP/M คำสั่งนี้ทำให้เราสามารถรู้ถึงไฟล์ต่าง ๆ ที่เก็บอยู่ในแผ่นดิสเก็ตต์ว่าแต่ละไฟล์ใช้หน่วยความจำเท่าไร และในแผ่นดิสเก็ตต์เหลือเนื้อที่ของหน่วยความจำอีกเท่าใด ก่อนการเรียกใช้โปรแกรม STAT เราจะต้องมั่นใจเสียก่อนในแผ่นดิสเก็ตต์ของเรามีโปรแกรม STAT COM อยู่ด้วยซึ่งการตรวจสอบทำได้โดยการใช้คำสั่ง DIR ดังต่อไปนี้คือ

A>DIR STAT.COM

ถ้าในแผ่นดิสเก็ตต์มีโปรแกรม STAT อยู่ด้วย เครื่องจะแสดงบนจอภาพว่า

STAT.COM

เป็นการบอกเราว่าในดิสเก็ตต์มีโปรแกรม STAT.COM อยู่ด้วย แต่ถ้าในแผ่นดิสเก็ตต์ไม่มีโปรแกรมนี้อยู่ เครื่องจะแสดงผลว่า NOT FOUND ดังนี้คือ

A > DIR STAT.COM

NOT FOUND

แต่อย่าเพิ่งตกใจเพราะโปรแกรม STAT.COM ของเราอาจจะอยู่ในดิสค์ไดร์ฟอื่น เช่น ในไดร์ฟ B: C: หรือ D: ดังนั้นเราอาจทำการตรวจสอบว่ามีโปรแกรม STAT.COM อยู่ในไดร์ฟอื่นหรือไม่ โดยการใส่ชื่อไดร์ฟที่ต้องการตรวจสอบนำหน้าชื่อโปรแกรม ตัวอย่างเช่น

A>DIR B:STAT.COM

หรือ A>DIR C:STAT.COM

เมื่อเราสามารถหาจนพบว่าโปรแกรม STAT.COM อยู่ในไดร์ฟใด เราต้องนำแผ่นดิสเก็ตต์จากไดร์ฟนั้นออกไปไว้ในไดร์ฟ A: ก่อนจึงจะสามารถเรียกใช้โปรแกรม STAT.COM ได้

STAT กับการตรวจสอบไฟล์ในดิสเก็ตต์

โปรแกรม STAT สามารถให้รายละเอียดต่าง ๆ เกี่ยวกับจำนวนของไฟล์ ขนาดของไฟล์ และชนิดของไฟล์ต่าง ๆ ในดิสเก็ตต์แก่เราได้ อาจกล่าวได้อีกนัยหนึ่งว่า STAT คือ โปรแกรมที่ทำหน้าที่ขยายส่วนการทำงานของ DIR อีกทีหนึ่ง นอกจากนี้ STAT ยังบอกแก่เราด้วยว่าในแผ่นดิสเก็ตต์มีที่ว่างอีกจำนวนเท่าใด

การเรียกใช้ STAT นั้นสามารถทำได้เช่นเดียวกับทรานเชียนต์คอมมานด์อื่น ๆ คือ เราสามารถใส่คำสั่งตามหลัง STAT ได้ โดยโปรแกรม STAT จะนำคำสั่งที่ใส่ตามลงไปนั้นไปแปลความหมายและทำตามคำสั่ง

คำสั่ง STAT ที่เกี่ยวข้องกับดิสเก็ตต์นั้นมีรูปแบบการเขียนคำสั่ง 5 แบบคือ

1. A > STAT

ถ้าเราเรียกใช้ STAT โดยไม่พิมพ์คำสั่งหรือข้อความตามหลัง STAT โปรแกรม STAT จะทำการแสดงจำนวนของหน่วยความจำที่มีเหลืออยู่ในแผ่นดิสเก็ตต์ออกมาทางจอภาพ ตัวอย่างเช่น

A > STAT

BYTES REM ING ON A: 185 K

บรรทัดที่สองคือข้อความซึ่งเครื่องจะแสดงออกมาเพื่อบอกแก่ผู้ใช้ว่ามีเนื้อที่เหลืออยู่เท่าใดในดิสเก็ตต์แผ่นที่เราต้องการรู้

2. A > STAT X:

รูปแบบคำสั่งนี้เป็นรูปแบบที่เครื่องจะแสดงผลเหมือนกับรูปแบบแรก คือ บอกจำนวนหน่วยความจำที่เหลือในแผ่นดิสเก็ตต์ แต่เราสามารถกำหนดได้ว่าต้องการให้แสดงจำนวนหน่วยความจำที่เหลือในไดร์ฟใด โดยการใส่ชื่อไดร์ฟแทนตัวแปร X นั่นเอง ตัวอย่างเช่น

A>STAT B:

BYTES REMAINING ON B: 156 K

3. A > STAT <file name. file type>

รูปแบบของคำสั่งนี้ คือ การใส่ชื่อและชนิดของไฟล์ที่ต้องการทราบขนาดลงในคำสั่ง STAT เครื่องจะแสดงรายละเอียดของจำนวนเรคอร์ดในไฟล์จำนวนไบท์ที่ใช้ในการเก็บในดิสค์เก็ตต์จำนวนบล็อก หรือที่เรียกว่าจำนวนเอ็กซ์เทนต์ (Extent) ของไฟล์ ตัวอย่างของการใช้คำสั่งและการแสดงผลแสดงดังต่อไปนี้คือ

```
A >STAT ASM.COM
RECS BYTES EX D:FILENAME.TYP
 75   36 K
           3 :
               A: ASM.COM
```

จำนวนเรคอร์ดของไฟล์ จำนวนไบท์ของไฟล์ จำนวนเอ็กซ์เทนต์ ดิสค์ไดร์ฟ ชื่อและชนิดของไฟล์

จากตัวอย่างข้างต้นเราต้องการดูสถานะของไฟล์ชื่อ ASM.COM CP/M จะแสดงรายละเอียดจำนวนเรคอร์ด จำนวนไบท์ จำนวนเอ็กซ์เทนต์และชื่อไฟล์พร้อมทั้งดิสค์ไดร์ฟที่ไฟล์อยู่ออกมา

ในกรณีที่เราต้องการดูสถานะของไฟล์ใดในดิสค์ไดร์ฟอื่น เราก็สามารถทำได้โดยการใส่ชื่อไดร์ฟนำหน้าชื่อไฟล์ ดังตัวอย่าง เช่น

```
A>STAT  B:PIP.COM
RECS BYTES EX D:FILENAME.TYP
13     8K     1   B: PIP.COM
```

4. A > STAT<* . file type>

หรือ A>STAT<filename.*>

หรือ A>STAT<* . *>

รูปแบบของคำสั่งนี้เป็นการดูสถานะของไฟล์ทีละหลาย ๆ ไฟล์พร้อม ๆ กัน รูปแบบแรกเป็นการดูสถานะของไฟล์ทุกไฟล์ที่มีชนิดของไฟล์ที่เรากำหนดเครื่องหมาย * ข้างหน้ามีความหมายว่าทุกไฟล์ ตัวอย่างเช่น

A STAT * . COM

เป็นการขอดูสถานะของไฟล์ทุกไฟล์ที่เป็นชนิดคอมมานด์ไฟล์ ซึ่งในกรณีนี้ไฟล์ทุกไฟล์ที่เก็บอยู่ในดิสเก็ตต์เป็นชนิดคอมมานด์ไฟล์คือมีนามสกุลของไฟล์เป็น .COM จะถูกแสดงรายละเอียดออกมา ดังตัวอย่างเช่น

```
A>STAT  *  COM
RECS   BYTES   EX   D:FILENAME.TYP
75     36K     3    A:ASM.COM
50     25K     2    A:DDT.COM
30     14K     1    A:ED.COM
 5      1K     1    A:LOAD.COM
13      8K     1    A:PIP.COM
25     12K     1    A:STAT.COM
```

สำหรับรูปแบบถัดมา เป็นการขอดูสถานะไฟล์ทุกชนิดที่มีชื่อเดียวกัน ตัวอย่างเช่น

A > STAT TEST *

RECS	BYTES	EX	D:FILENAME.TYP
50	25K	2	A:TEST.BAS
30	14K	1	A:TEST.COM
50	25K	2	A:TEST.BAK

จากตัวอย่างนี้ไฟล์ทุกชนิดที่มีชื่อเดียวกันคือ TEST ซึ่งประกอบด้วย TEST.BAS TEST.BAK และ TEST.COM จะถูกนำรายละเอียดของสถานะของไฟล์ออกมาแสดง

ในรูปแบบสุดท้ายเป็นการขอดูสถานะของไฟล์ทุกไฟล์ที่มีอยู่ในแผ่นดิสเก็ตต์ในกรณีนี้เมื่อ CP/M แสดงสถานภาพของไฟล์ทุกไฟล์แล้วจะบอกจำนวนของหน่วยความจำที่เหลืออยู่ในแผ่นดิสเก็ตต์ออกมาด้วยดังตัวอย่างที่แสดงต่อไปนี้

A>STAT * · *

RECS	BYTES	EX	D:FILENAME.TYP
75	36K	3	A:ASM.COM
13	8K	1	A:ASM.COM
25	12K	1	A:STAT.COM
10	5K	1	A:DUMP.ASM
50	25K	2	A:TEST.BAS
50	25K	2	A:TEST.BAK

BYTES REMAINING ON A: 170K

ทำนองเดียวกันกับการใช้ STAT ในรูปแบบอื่น ๆ การขอดูสถานภาพของไฟล์ในดิสเก็ตต์ซึ่งอยู่ในดิสค์ไดร์ฟอื่นสามารถทำได้โดยการเพิ่มชื่อของไดร์ฟลงหน้าชื่อของไฟล์ที่ต้องการ ตัวอย่างเช่น

A > STAT B: * . COM

หรือ A> STAT B: TEST . *

หรือ A> STAT B: * · *

5. A > STAT X: = R/O

รูปแบบคำสั่ง STAT ที่เกี่ยวข้องกับดิสเก็ตต์คำสั่งสุดท้ายคือ การกำหนดให้ดิสค์ไดร์ฟใดทำหน้าที่อ่านได้เท่านั้น (READ - ONLY) คือจะเขียนข้อมูลลงในดิสค์ไดร์ฟนั้นไม่ได้ ตัวอย่างเช่น

A>STAT B: = R/O

กรณีนี้ดิสค์ไดร์ฟ B: จะถูกกำหนดให้อ่านข้อมูลจากดิสเก็ตต์ได้แต่เพียงอย่างเดียวจะทำการเขียนข้อมูลลงในดิสเก็ตต์ไม่ได้ ถ้าเราพยายามเขียนข้อมูลในดิสเก็ตต์ที่อยู่ในไดร์ฟ B: เครื่องจะบอกว่าเกิดความผิดพลาดด้วยความจำ

BDOS ERROR ON B: READ ONLY

การใช้ STAT รูปแบบนี้มีวัตถุประสงค์หลักเพื่อป้องกันการเขียนข้อมูลลงในแผ่นดิสเก็ตต์ที่เราเก็บ

ข้อมูลที่สำคัญมากไว้และไม่ต้องการให้มีการเขียนทับหรือลบข้อมูลโดยบังเอิญ

ถ้าต้องการยกเลิกสถานะความเป็น R/O ของดิสค์ไดร์ฟที่ทำการ STAT ไว้สามารถทำได้โดยการทำรีบูท (Rebooting) ระบบเสียใหม่ด้วยการกดคีย์ CTRL และ C พร้อม ๆ กัน

STAT กับการเลือกอุปกรณ์ใช้งาน

STAT สามารถทำให้ผู้ใช้เครื่องไม่ได้กำหนดหรือเลือกใช้อุปกรณ์ต่าง ๆ ได้ตามที่ต้องการดังได้-กล่าวมาแล้วในบทที่ 2 ว่า CP/M อนุญาตให้เราใช้ชื่อลอจิกคัล ดีไวซ์ (Logical Device) เพื่อกำหนดให้แก่อุปกรณ์อินพุต/เอาต์พุตได้ตามที่เราต้องการ

ชื่ออุปกรณ์ลอจิกหรือลอจิกคัลดีไวซ์ของ CP/M ซึ่งสามารถกำหนดให้แก่อุปกรณ์อินพุต/เอาต์พุตได้มีอยู่ 4 ชื่อ CONSOLE (CON:) LIST (LST:) READER (RDR:) และ PUNCH (PUN:) ซึ่งอุปกรณ์ลอจิกที่ใช้กันค่อนข้างมากคือ CONSOLE และ LISTดังนั้นเราจะอธิบายรายละเอียดสำหรับการใช้อุปกรณ์ CON: และ LST ในหัวข้อนี้

การที่อุปกรณ์อินพุต/เอาต์พุตใดจะทำงานได้นั้นจะต้องมีโปรแกรมสั่งให้อุปกรณ์นั้นทำงานโปรแกรมหรือที่เราเรียกว่าไดรเวอร์ (driver) เหล่านี้เป็นส่วนโปรแกรมย่อยภายใน BIOS ดังนั้นเมื่อ CP/M มีโปรแกรมไดรเวอร์เพื่อสั่งให้อุปกรณ์อินพุต/เอาต์พุตต่าง ๆ ทำงานได้ เราก็สามารถเลือกอุปกรณ์ที่ต้องการโดยกำหนดชื่ออุปกรณ์ลอจิกชื่อใดชื่อหนึ่งให้แก่อุปกรณ์ที่ต้องการใช้งาน ตัวอย่างเช่นถ้าใน BIOS ของ CP/M มีโปรแกรมไดรเวอร์อยู่ 3 โปรแกรมคือของจอภาพ CRT และเครื่องพิมพ์ และของโมเด็ม (Modem) เราอาจกำหนดชื่อ CON: ให้แก่จอภาพ CRT หรือโมเด็ม และ LST:ให้แก่จอภาพ CRT เครื่องพิมพ์หรือโมเด็มได้

CP/M มีชื่อมาตรฐานซึ่งสามารถกำหนดให้แก่อุปกรณ์อินพุต/เอาต์พุตอยู่หลายชื่อดังแสดงต่อไปนี้คือ

TTY : อุปกรณ์เทเลไทป์ (teletype) (คอนโซลความเร็วต่ำ)
CRT : อุปกรณ์จอภาพ CRT (คอนโซลความเร็วสูง)
BAT : การประมวลผลแบบแบทซ์ (Batch processing)
UCL : คอนโซลซึ่งกำหนดโดยผู้ใช้ (User defined console)
PTR : เครื่องอ่านเทปกระดาษ (Paper tape reader)
UR1 : เครื่องอ่านข้อมูลกำหนดโดยผู้ใช้ #1 (User defined reader#1)
UR2 : เครื่องอ่านข้อมูลกำหนดโดยผู้ใช้ #2 (User defined reader#2)
PTP : เครื่องเจาะเทปกระดาษ (Paper tape punch)
UP1 : เครื่องเจาะกำหนดโดยผู้ใช้ #1 (User defined punch#1)
UP2 : เครื่องเจาะกำหนดโดยผู้ใช้ #2 (User defined punch#2)
LPT : เครื่องพิมพ์ผล (Line printer)
UL1 : เครื่องพิมพ์ผลกำหนดโดยผู้ใช้ (User defined list device)

อย่างไรก็ตามชื่อมาตรฐานต่าง ๆ เหล่านี้เป็นเพียงชื่อที่กำหนดเป็นดัมมี (dummy) เท่านั้น

ตัวอย่างเช่นโปรแกรมไดร์ฟเวอร์ที่กำหนดให้มีชื่อเป็น LST: หรือเครื่องพิมพ์ผลโดยแท้ที่จริงอุปกรณ์ที่ไดรเวอร์ทำการควบคุมการทำงานจริงอาจไม่ใช่เครื่องพิมพ์ผลแต่อาจเป็นโมเด็มซึ่งสามารถนำไปเชื่อมต่อกับอุปกรณ์แสดงผลอื่น ข้อจำกัดที่สำคัญประการเดียวในการใช้อุปกรณ์ก็คืออุปกรณ์ที่ทำการต่ออยู่หรือใช้จริงจะต้องมีคุณลักษณะเหมือนกับชื่อของอุปกรณ์ที่กำหนดโดย CP/M เช่นไดรเวอร์ซึ่งทำหน้าที่ควบคุมการทำงานของอุปกรณ์ที่เป็นได้เฉพาะเอาต์พุตแต่เพียงอย่างเดียว เช่นเครื่องพิมพ์ผล จะกำหนดให้ควบคุมการทำงานของอุปกรณ์ซึ่งสามารถเป็นได้ทั้งอินพุตและเอ้าต์พุตเช่น จอภาพ CRT ไม่ได้

เนื่องจากชื่ออุปกรณ์มาตรฐานของ CP/M มีอยู่ถึง 12 ชื่อ การจำชื่อต่าง ๆ เหล่านี้คงไม่ใช่เรื่องง่ายนัก เพื่อช่วยให้ผู้ใช้สามารถเรียกใช้ชื่อของอุปกรณ์ต่าง ๆ ได้ถูกต้อง CP/M อนุญาตให้ผู้ใช้ทำการเรียกชื่อของอุปกรณ์ต่าง ๆ ออกมาดูได้โดยคำสั่ง STAT VAL: ดังแสดงต่อไปนี้

```
A>STAT VAL:
CON: = TTY: CRT: BAT: UC1:
RDR: = TTY: PTR UR1: UR2:
PUN: = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:
```

รายละเอียดที่เราได้จาก CP/M เมื่อเรียกใช้คำสั่ง STAT VAL: คือชื่ออุปกรณ์มาตรฐานของ CP/M ซึ่งสามารถกำหนดให้แก่ลอจิกดีไวซ์ CON: เป็น TTY: CRT: BAT: หรือ UC1 ได้แต่จะกำหนดให้ CON: เป็น LPT: ไม่ได้

ผู้ใช้เครื่องคอมพิวเตอร์อาจทำการเปลี่ยนแปลงอุปกรณ์ต่าง ๆ ที่กำลังใช้อยู่โดยเปลี่ยนเป็นอุปกรณ์อื่นได้ โดยการกำหนดชื่ออุปกรณ์ให้แก่ลอจิกดีไวซ์ตามหลังคำสั่ง STAT ตัวอย่างเช่น

```
A>STAT CON: = CRT
```

คำสั่งนี้เป็นการกำหนดให้คอนโซลเป็นจอภาพ CRT และแป้นพิมพ์

การสั่งให้ CP/M แสดงสถานภาพของอุปกรณ์ที่ต่ออยู่กับลอจิกดีไวซ์ในขณะใช้งานอยู่ทำได้โดยการใช้คำสั่ง STAT DEV: ดังตัวอย่างต่อไปนี้

```
A>STAT DEV:
CON: = CRT:
RDR: = TTY:
PUN: = TTY:
LST: = TTY:
```

ในรายละเอียดที่แสดงด้วยคำสั่ง STAT DEV: ข้างต้นจะเห็นได้ว่าในขณะนี้เรากำหนดให้คอนโซลเป็นจอภาพ CRT ส่วน READER PUNCH และ LIST ถูกกำหนดให้เป็นเทเลไทป์ซึ่งถ้าเราต้องการกำหนดให้ LST: เป็นเครื่องพิมพ์ผล (LPT:) เราก็สามารถทำได้ด้วยคำสั่ง STAT LST: = LPT: ดังแสดงต่อไปนี้

```
A>STAT LST: = LPT:
A>STAT DEV:
CON: = CRT:
```

RDR: = TTY:

PUN: = TTY:

LST: = LPT:

ทั้งหมดนี้เป็นเรื่องราวของการใช้คำสั่ง STAT เมื่อได้ใช้ CP/M เราจะพบว่าจะมีการเรียกใช้ STAT อยู่บ่อยทีเดียว

ความรู้ทั่วไปเกี่ยวกับ PIP

PIP เป็นโปรแกรมที่มีชื่อย่อมาจาก Peripheral Interchange Program ซึ่งมีหน้าที่ในการก๊อบปี้ไฟล์จากดิสเก็ตต์หนึ่งไปยังดิสเก็ตต์อีกแผ่นหนึ่ง PIP เป็นคำสั่งประเภททรานเชียนต์คอมมานด์เช่นเดียวกันกับ STAT และเป็นคำสั่งที่มีการใช้งานกันบ่อยมากที่สุดคำสั่งหนึ่ง นอกจากการก๊อบปี้ไฟล์จากดิสเก็ตต์หนึ่งไปไว้ยังอีกดิสเก็ตต์หนึ่ง PIP ยังสามารถทำอะไรได้อีกหลายอย่าง เช่น การนำเอาข้อมูลในไฟล์ไปออกที่อุปกรณ์ลอจิก เช่น LST: เป็นต้น แต่เนื่องจากประมาณ 99% ของการใช้งาน PIP เป็นการใช้เพื่อทำการก๊อบปี้ไฟล์ ในที่นี้จึงจะขอกล่าวถึงเน้นเฉพาะการใช้คำสั่ง PIP เพื่อการก๊อบปี้ไฟล์ ส่วนการใช้งานอื่นจะกล่าวถึงแต่พอสังเขป

การเรียกใช้คำสั่ง PIP สามารถทำได้โดยการพิมพ์คำสั่งตามคำสั่ง PIP หรือไม่ก็ได้ ถ้าเราใส่คำสั่งหรือข้อความตามหลังคำสั่ง PIP ในบรรทัดเดียวกันเครื่องคอมพิวเตอร์จะทำการโหลดโปรแกรม PIP .COM เข้ามาในหน่วยความจำ ทำการเอ็กซิคิวคำสั่งหลัง PIP เสร็จแล้วจะกลับเข้าไปยัง CCP และให้สัญลักษณ์ A> ออกมาบนจอภาพ แต่ถ้าเราไม่ใส่คำสั่งตามคำสั่ง PIP เมื่อกดคีย์ RETURN เครื่องคอมพิวเตอร์จะทำการโหลดโปรแกรม PIP.COM เข้ามาในหน่วยความจำแล้วจะขึ้นสัญลักษณ์ * เพื่อเตรียมรับคำสั่งที่ต้องการให้โปรแกรม PIP ทำต่อไป การป้อนคำสั่งทำได้โดยการคีย์คำสั่งเข้าไปทีละคำสั่งจนหมดเมื่อต้องการให้โปรแกรม PIP ทำการเอ็กซิคิวคำสั่งที่ใส่เข้าไปก็ทำได้โดยการกดคีย์ RETURN โดยไม่ต้องใส่ข้อความใด ๆ เมื่อจอภาพแสดงเครื่องหมายดอกจัน * ในบรรทัดใหม่ เมื่อได้ทำการเอ็กซิคิวคำสั่งต่าง ๆ แล้ว จะกลับเข้าใน CCP และแสดงสัญลักษณ์ A เพื่อเตรียมพร้อมที่จะรับคำสั่งต่าง ๆ ของ CP/M ต่อไป

รูปแบบทั่วไปของคำสั่ง PIP คือ

ชื่อดิสค์ไดร์ฟ

D:FILE NAME.TYP = D:FILE NAME.TYP

ชื่อและชนิดของไฟล์ที่ต้องการก๊อบปี้ไปไว้ ชื่อและชนิดของไฟล์ต้นฉบับ

ตัวอย่าง เช่น ถ้าเราต้องการก๊อบปี้โปรแกรม STAT.COM จากแผ่นดิสเก็ตต์ต้นฉบับในไดร์ฟ A: ไปไว้ในไดร์ฟ B: รูปแบบคำสั่งซึ่งเราสามารถใช้ได้คือ

A>PIP B: STAT.COM = A: STAT.COM

รูปแบบนี้เป็นการเขียนคำสั่งหลังคำสั่ง PIP หรือเราอาจใช้รูปแบบซึ่งเขียนคำสั่งต่างหากได้ดังนี้คือ

A>PIP

* B: STAT.COM = A: STAT.COM

* คีย์ return

รูปแบบคำสั่งทั้งสองจะให้ผลการทำงานเหมือนกันคือ ก๊อบปี้โปรแกรม STAT.COM จากดิสเก็ตต์ในไดร์ฟ A: ไปไว้ในไดร์ฟ B:

ในกรณีที่เราต้องการก๊อบปี้ไฟล์จากดิสเก็ตต์หนึ่งไปยังอีกดิสเก็ตต์หนึ่งโดยไม่จำเป็นต้องเปลี่ยนชื่อหรือต้องการคงชื่อเดิมของไฟล์ไว้เราสามารถใช้รูปแบบคำสั่งคือ

A>PIP B: = A: STAT.COM

นอกจากนี้เรายังสามารถก๊อบปี้ไฟล์ได้ทีละหลาย ๆ ไฟล์ หรือทุกไฟล์โดยใช้วิธีการกำหนดชื่อไฟล์หรือชนิดของไฟล์ด้วยเครื่องหมายดอกจัน * ได้ดังตัวอย่างเช่น

A>PIP B: = A: * . COM

หรือ A>PIP B: = A: * TEXT . *

หรือ A>PIP B: = A: * . *

ระบบการใช้คำสั่ง PIP อื่น ๆ

คำสั่ง PIP ยังอนุญาตให้เราทำการเปลี่ยนชื่อไฟล์หรือชนิดของไฟล์ได้ในขณะทำการก๊อบปี้โดยเพียงแต่ใส่ชื่อใหม่และชนิดของไฟล์ที่ต้องการเปลี่ยนไว้ทางขวาของเครื่องหมายเท่ากับในคำสั่ง PIP ตัวอย่างเช่น สมมติเราต้องการเก็บไฟล์สำรองหรือแบคอัพไฟล์ (Backup file) ไว้ในดิสค์ไดร์ฟ B: โดยเก็บไว้ในไฟล์ชนิดแบคอัพคือ BAK เราสามารถเขียนคำสั่ง PIP ได้คือ

A>PIP B: CPMBOOK.BAK = A: CPMBOOK.TXT

จะเห็นว่าในกรณีนี้ไฟล์ต้นฉบับเป็นชนิดเท็กซ์คือ .TXT เมื่อเราทำการก๊อบปี้มาไว้ในดิสค์ไดร์ฟ B: จะถูกเปลี่ยนเป็นไฟล์ชนิด .BAK

เนื่องจากการใช้ PIP ส่วนใหญ่จะใช้ในการก๊อบปี้ไฟล์เพื่อเก็บไว้เป็นไฟล์สำรองเพื่อให้เราแน่ใจว่าไฟล์ที่ก๊อบปี้ได้นั้นถูกต้องตรงกันกับไฟล์ต้นฉบับจริง คำสั่ง PIP มีรูปแบบเพื่อให้เราสามารถตรวจสอบการก๊อบปี้ได้โดยการพิมพ์อักษร V ภายในวงเล็บใหญ่ คือ V หลังคำสั่งการก๊อบปี้ไฟล์ในรูปแบบคำสั่งนี้ CP/M จะทำการตรวจทาน (Verify) ผลการก๊อบปี้ไฟล์โดยหลังจากการก๊อบปี้ไฟล์เสร็จแล้ว เครื่องจะทำการอ่านไฟล์จากดิสเก็ตต์ต้นฉบับและดิสเก็ตต์ที่ก๊อบปี้ไปไว้ออกมาเปรียบเทียบกัน ถ้ามีข้อความหรือส่วนใดที่ไม่ตรงกันเครื่องก็จะแสดงผลออกมาทางจอภาพ ตัวอย่างการใช้คำสั่ง PIP รูปแบบนี้แสดงดังต่อไปนี้

A PIP

* B: = A: * . COM V

* B: STAT.COM = A: STAT.COM V

* B: * . BAK = A: * . DAT V

* คีย์ return

คำสั่งแรกใน PIP เป็นการก๊อบปี้ไฟล์ทุกไฟล์ที่เป็นชนิด .COM จากดิสค์ไดร์ฟ A: ไปไว้ในไดร์ฟ B: แล้วที่การตรวจทานว่ามีอะไรบกพร่องหรือไม่ คำสั่งที่สองเป็นการก๊อบปี้โปรแกรม STAT. COM

จากไดร์ฟ A: ไปไว้ในไดร์ฟ B: พร้อมทั้งทำการตรวจสอบการก๊อบปี้ และคำสั่งสุดท้ายเป็นการก๊อบปี้ไฟล์ทุกไฟล์ที่เป็นชนิดไฟล์ข้อมูล คือ .DAT จากไดร์ฟ A: ไปไว้ในไดร์ฟ B: โดยเปลี่ยนเป็นชนิดแบคอัพไฟล์ .BAK แต่ยังคงชื่อไฟล์เดิมไว้ พร้อมทั้งทำการตรวจทานหลังการก๊อบปี้

การแสดงผลด้วยคำสั่ง PIP

นอกเหนือจากการใช้คำสั่ง PIP ในการก๊อบปี้ไฟล์แล้ว เรายังสามารถใช้คำสั่ง PIP ในการแสดงผลข้อมูลของไฟล์หรือโปรแกรมในไฟล์ได้อีกด้วย ตัวอย่างเช่น

A>PIP LST: = COPY.TXT

คำสั่งนี้เป็นการสั่งให้เครื่องคอมพิวเตอร์ทำการนำไฟล์ที่ชื่อว่า COPY.TXT ออกพิมพ์ผลทาง LST: หรือเครื่องพิมพ์

การนำไฟล์ออกแสดงผลด้วยคำสั่ง PIP มีข้อดีที่ว่าเราสามารถสั่งให้เครื่องคอมพิวเตอร์ทำการแสดงผลหรือพิมพ์ผลทีละหลาย ๆ ไฟล์ได้โดยคำสั่งเดียวโดยอาศัยรูปแบบการกำหนดชื่อไฟล์ด้วยเครื่องหมายดอกจัน เช่นเดียวกับที่เคยกล่าวมาแล้ว ตัวอย่างเช่น

A>PIP LST: = * . TXT
หรือ A>PIP LST: = B:COPY.*
หรือ A>PIP LST: = B: * · *

คำสั่งแรกเป็นการสั่งให้พิมพ์ไฟล์ทุกไฟล์เป็นชนิด .TXT จากไดร์ฟ A: ออกที่เครื่องพิมพ์ สังเกตว่าเราไม่ต้องกำหนดไดร์ฟด้วยอักษร A: ทั้งนี้เพราะดิสค์ไดร์ฟที่กำลังทำงานอยู่คือไดร์ฟอยู่แล้ว คำสั่งที่สองเป็นการสั่งให้พิมพ์ไฟล์ทุกไฟล์ที่มีชื่อว่า COPY จากไดร์ฟ B: ออกทางเครื่องพิมพ์ ส่วนคำสั่งสุดท้ายเป็นการสั่งให้พิมพ์ไฟล์ทุกไฟล์จากไดร์ฟ B: ออกทางเครื่องพิมพ์

จากคำสั่ง STAT และ PIP ที่ได้กล่าวมาแล้วเราคงเห็นแล้วว่าทั้งสองคำสั่งนี้มีประโยชน์มากเพียงใด และเมื่อผนวกคำสั่งทั้งสองเข้ากับคำสั่ง FORMAT ดังได้กล่าวมาในบทก่อนหน้านี้ เราก็สามารถก๊อบปี้ดิสเก็ตต์เก็บไว้เป็นแผ่นสำรองในการใช้งานได้มากมายตามที่ต้องการ

6

ED โปรแกรมของ CP/M

ถึงแม้ว่าจะมีไฟล์อยู่หลายชนิดที่ใช้ใน CP/M (.COM,.BAS, .TXT) แต่ไฟล์ที่เราสามารถเข้าไปเกี่ยวข้องภายในไฟล์เพื่อแก้ไขข้อความหรือสร้างไฟล์ขึ้นมาใหม่นั้นโดยมากเป็นพวก ไฟล์เอกสาร (text) ซึ่งเป็นไฟล์ที่สร้างขึ้นโดยตัวอักขระทุกตัวที่มีอยู่บนคีย์บอร์ด (ตัวเลข ตัวอักษร สัญลักษณ์พิเศษ ได้แก่ เครื่องหมายในการคำนวณ +, −, *, / เครื่องหมายเปรียบเทียบ < , =, > และเครื่องหมายอื่น ๆ เช่น $, ., ( ) เป็นต้น) ไฟล์เอกสารที่สร้างขึ้นนี้สร้างโดย โปรแกรมที่เรียกว่า เอดิเตอร์ (Editor) เอดิเตอร์โปรแกรมเป็นโปรแกรมที่คล้ายกับเครื่องพิมพ์ดีดอิเล็กทรอนิกส์ แต่เป็นการพิมพ์ลงในหน่วยความจำของคอมพิวเตอร์แทนที่จะพิมพ์ลงบนกระดาษพิมพ์

ED ที่อยู่บนแผ่น CP/M เป็นเอดิเตอร์ของ CP/M ที่ใช้ในการสร้างและแก้ไขซอร์ส ไฟล์สำหรับตัวแปลภาษา (Assembler) ของภาษาแอสเซมบลี้ ภาษาเบสิค และภาษาอื่น ๆ ซึ่งต้องการซอร์ส โปรแกรมที่เขียนด้วยภาษานั้น ๆ เพื่อทำการแปลเป็นภาษาเครื่อง (machine code) ซึ่งเป็นภาษาที่คอมพิวเตอร์สามารถเข้าใจได้(.COM) ED สามารถให้การป้อนข้อมูล แสดงผลและแก้ไขไฟล์เอกสารภายใต้การทำงานของ CP/M ดังนั้นไฟล์ที่สร้างขึ้นจะมีรูปแบบ (Format) ที่แน่นอนและยังสามารถใช้ในการแก้ไขซอร์สไฟล์ของภาษาฟอร์แทรน และภาษาอื่น ๆ รวมทั้งไฟล์เพื่อการพิมพ์ (.PRN) ซึ่งสร้างโดยโปรแกรม ASM

ED ทำหน้าที่เป็นตัวที่ทำการแก้ไข (text editor) ข้อความเฉพาะบรรทัดที่เราระบุไว้ได้เท่านั้น (line oriented) ซึ่งต่างจากเอดิเตอร์อีกแบบหนึ่งซึ่งสามารถแก้ไขข้อความบรรทัดใดก็ได้ที่ปรากฏบนจอทีวีโดยการเลื่อนตัวชี้ตำแหน่ง (CURSOR) ไปยังบรรทัดต่าง ๆ แล้วทำการแก้ไขในกรณีหลังนี้เราเรียกว่าฟูลสกรีนเอดิเตอร์ นอกจากนี้เรายังมีเวิร์ดโปรเซสเซอร์ที่ใช้ในการสร้างไฟล์เอกสารเช่นกัน เวิร์ดโปรเซสซิ่ง (Word processing) จะพิมพ์ข้อความลงบนจอทีวีเหมือนกับที่เราพิมพ์ดีดโดยเครื่องพิมพ์ดีดลงบนกระดาษ รวมทั้งการจัดระยะซ้ายขวา การเขียนเส้นใต้ เลขที่หน้า หัวกระดาษ ลงท้าย และการทำงานอื่น ๆ ที่ออกแบบไว้

ED โปรแกรมใน CP/M ไม่สามารถทำได้เหมือนเวิร์ดโปรเซสเซอร์แต่จะใช้หลักการพื้นฐานที่จะเป็นในการสร้างและปรับปรุงไฟล์เอกสาร ดังนั้นถ้าเราต้องการให้คอมพิวเตอร์ทำงานลักษณะแบบเวิร์ดโปรเซสซิ่ง ก็ต้องซื้อโปรแกรมสำเร็จรูป (package) ซึ่งมีให้เลือกมากมายในท้องตลาด

วิธีใช้ ED

ED โปรแกรมเรียกใช้ได้โดยการพิมพ์ตัวอักษร ED แล้วตามด้วยชื่อไฟล์ ชนิดของไฟล์ เมื่อกดคีย์ RETURN แล้วโปรแกรม ED จะได้รับการเรียกมาจากดิสค์ ในกรณีไม่มีชื่อไฟล์ปรากฏอยู่บนดิสค์ ED โปรแกรมจะถือว่าเป็นไฟล์ที่สร้างขึ้นมาใหม่ และพิมพ์อักษร NEW FILE บนจอทีวีบอกให้เรารู้ ดังตัวอย่างข้างล่างนี้

A > ED FILENAME. FILETYPE

NEW FILE

*

การที่พูดว่า ED เป็นโปรแกรมสร้างไฟล์ขึ้นมาใหม่นั้นหมายความว่า CP/M จะมีไฟล์นี้อยู่บนดิสค์ และชื่อไฟล์จะอยู่ในดิสค์ ไดเรกทอรี ในกรณีที่สร้างขึ้นใหม่ยังไม่มีอยู่ในดิสค์

เมื่อ ED พร้อมจะรับคำสั่งจะแสดง "*" ขึ้นบนจอดังตัวอย่างข้างบน ในกรณีที่ชื่อของไฟล์มีอยู่ในดิสค์แล้วการทำงานของ ED ก็สามารถอธิบายได้ดังรูปที่ 1 โดย ED โปรแกรมจะอ่านข้อมูลจากซอร์สไฟล์บนดิสค์มายังหน่วยความจำสำรอง เอกสารที่อยู่ในหน่วยความจำสำรองสามารถได้รับแก้ไขเปลี่ยนแปลงได้โดยไม่ต้องแก้ไขเอกสารข้อความในดิสค์

เอกสารที่สร้างขึ้นหรือที่แก้ไขในหน่วยความจำสำรองอาจจะเขียนลงบนไฟล์ชั่วคราวเมื่อใดก็ได้โดยผู้ใช้เป็นผู้สั่งในขณะการทำงานของ ED หลังจากสิ้นสุดการแก้ไขข้อความในหน่วยความจำสำรองและเอกสารส่วนที่ยังไม่ได้อ่านเข้ามาบนดิสค์จะได้รับการเขียนเข้าไปในไฟล์ชั่วคราว FILE.$$$ ซอร์สไฟล์เดิมจะได้รับการเปลี่ยนชื่อเป็น FILE . BAK และไฟล์ชั่วคราวจะได้รับการเปลี่ยนชื่อเป็น FILE . TXT ทั้งนี้เพื่อเป็นการป้องกันความผิดพลาดที่อาจเกิดขึ้น เราก็สามารถเปลี่ยนชื่อ FILE . BAK เป็น FILE . TXT แล้วใช้ ED โปรแกรมแก้ไขต่อไปหลังจากคำสั่ง ERA ลบโปรแกรมที่ผิดดังตัวอย่าง

A > ERA SORT.FOR ลบโปรแกรมที่ผิดหลังจากใช้ ED โปรแกรม

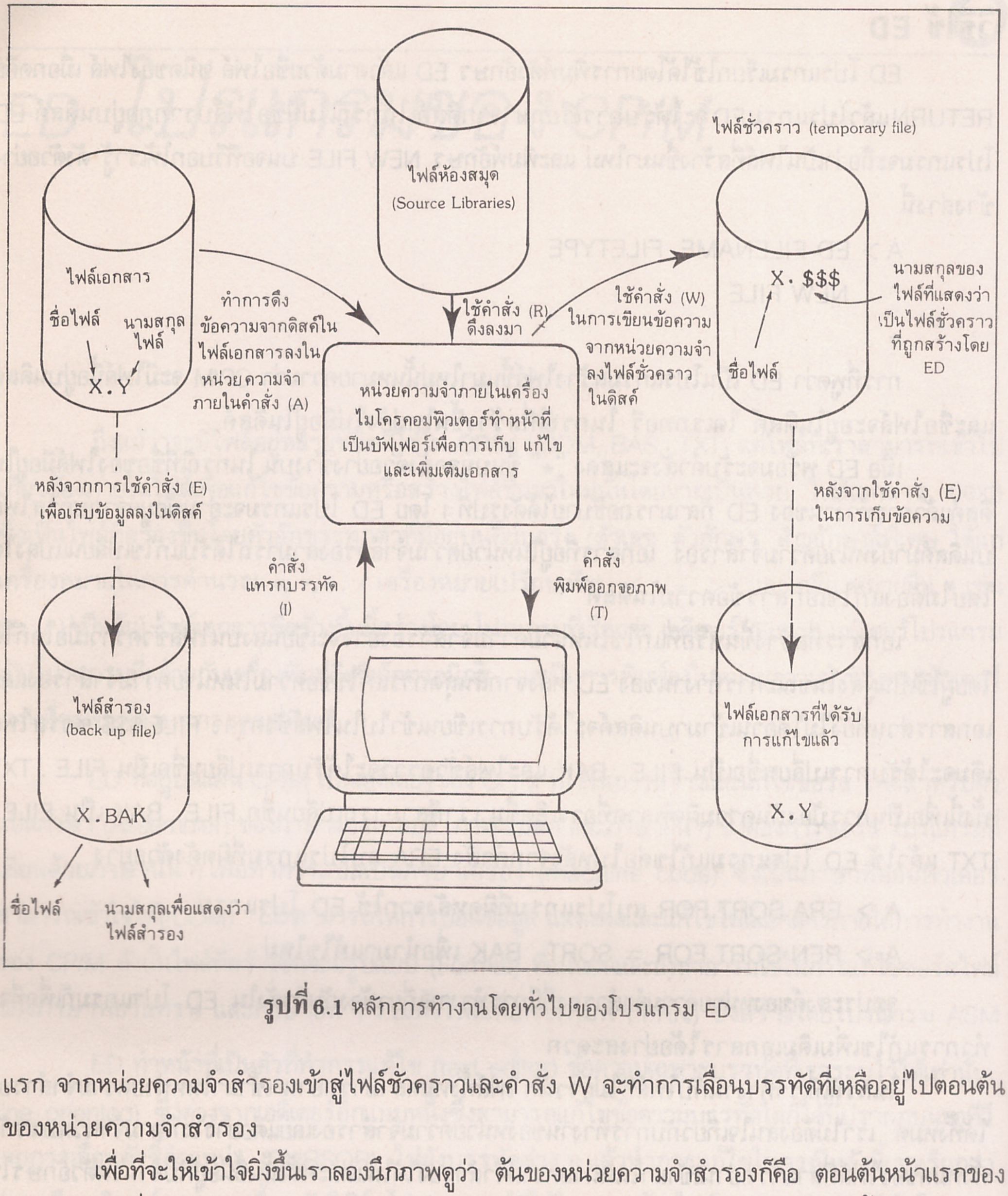
A > REN SORT.FOR = SORT . BAK เพื่อนำมาแก้ไขใหม่

จุดประสงค์ของหน่วยความจำสำรองที่มีการทำงานเกี่ยวข้องกับคำสั่งใน ED โปรแกรมก็เพื่อที่จะทำการแก้ไขเพิ่มเติมเอกสารได้อย่างสะดวก

ในกรณีที่เราทำงานกับไฟล์ที่ไม่ยาวนัก ไฟล์ทั้งหมดสามารถอ่าน เข้ามาในหน่วยความจำสำรองได้ทั้งหมด เราไม่ต้องสนใจเกี่ยวกับการทำงานของหน่วยความจำสำรองเลยแต่อย่างไรก็ตามจำนวนบรรทัดทั้งหมดของไฟล์ที่สามารถอ่านเข้ามาในหน่วยความจำสำรองได้ในแต่ละครั้งนั้นขึ้นอยู่กับจำนวนตัวอักษรในแต่ละบรรทัดและขนาดของหน่วยความจำสำรองของเครื่องคอมพิวเตอร์

ดังนั้นถ้าขนาดของเอกสารใหญ่กว่าขนาดของหน่วยความจำสำรองในดิสค์จะอ่านและเขียนบรรทัดที่เพิ่มกับหน่วยความจำสำรอง

คำสั่งใน ED สองคำสั่งที่ใช้ในการส่งผ่านเอกสารระหว่างไฟล์หลัก ไฟล์ชั่วคราวและหน่วยความจำสำรองคือคำสั่ง APPEND (A) และ WRITE (W) เช่น nA หมายถึงต้องการอ่าน n บรรทัดต่อมาจากซอร์สไฟล์เข้ามายังหน่วยความจำสำรองมาต่อท้ายเอกสารที่มีอยู่ในหน่วยความจำสำรอง ในกรณีที่เราอ่านเอกสารส่วนหนึ่งจากซอร์สไฟล์มาแก้ไข และ nW เป็นการเขียนข้อความจากหน่วยความจำสำรอง n บรรทัด



รูปที่ 6.1 หลักการทำงานโดยทั่วไปของโปรแกรม ED

แรก จากหน่วยความจำสำรองเข้าสู่ไฟล์ชั่วคราวและคำสั่ง W จะทำการเลื่อนบรรทัดที่เหลืออยู่ไปตอนต้นของหน่วยความจำสำรอง

เพื่อที่จะให้เข้าใจยิ่งขึ้นเราลองนึกภาพดูว่า ต้นของหน่วยความจำสำรองก็คือ ตอนต้นหน้าแรกของหนังสือเล่มหนึ่งและตอนท้ายของหน่วยความจำสำรองก็คือหน้าสุดท้ายของหนังสือเล่มนั้น

ในทั้งสองกรณี n จะแทนค่าตัวเลขระหว่าง 1 ถึง 65535 และเราอาจใช้เครื่องหมาย # แทน n โดย # จะถือเสมือนว่ามีค่าเท่ากับ 65535 หรือแทนจำนวนบรรทัดทั้งหมดซึ่งใช้ในกรณีอ่านไฟล์เข้าสู่หน่วยความจำสำรองหรือเขียนข้อความในหน่วยความจำสำรองลงสู่ไฟล์ชั่วคราวเมื่อเราไม่ทราบว่าไฟล์มีความยาวกี่บรรทัด แต่รู้ว่าสามารถบรรจุในหน่วยความจำสำรองได้หมด เช่น #A ใช้เพื่ออ่านไฟล์ต่อจากที่อ่านมาแล้วหรือเป็นการเพิ่มข้อความจากไฟล์ทั้งหมดลงสู่หน่วยความจำสำรองและ # W เป็นการเขียนข้อความจากหน่วยความจำสำรองลงสู่ไฟล์ชั่วคราวต่อจากบรรทัดที่เคยเขียนก่อนหน้านี้หรือเพิ่งเป็น

การเริ่มเขียนข้อมูลจากหน่วยความจำสำรองทั้งหมดลงสู่ไฟล์ชั่วคราว ถ้าข้อมูลในไฟล์ที่อ่านเข้ามามีขนาดใหญ่กว่าที่หน่วยความจำสำรองจะรับได้ ED จะบอกความผิดพลาดนี้ทางจอและเราสามารถใช้คำสั่งของ ED ต่อไป โดยไม่กำหนดให้อ่านข้อมูลเข้ามาทั้งหมดแต่ให้อ่านเข้ามาบางส่วน (เช่นเดียวกับการเขียนข้อมูลบางส่วนจากหน่วยความจำสำรองลงสู่ไฟล์ชั่วคราว)

ถ้า n เว้นไว้ ED จะตีความว่า n = 1 ดังนั้นคำสั่ง A และ W จะเขียนหรืออ่านเพียง 1 บรรทัดของเอกสารตามลำดับ

หากจะกล่าวเพิ่มเติมเกี่ยวกับซอร์สไฟล์ (Source file) ไฟล์ชั่วคราว หน่วยความจำสำรอง ED ส่วนประกอบที่สำคัญตัวหนึ่งคือ ตัวชี้ตำแหน่ง (CP) ED จะถือว่าไฟล์มีข้อมูลเป็นบรรทัดต่อเนื่องกันไปตลอดหน่วยความจำสำรอง แต่ละบรรทัดแยกโดยรหัส 0D ซึ่งหมายถึง return (Carriage return) ผู้ใช้สามารถเลื่อนตัวชี้ตำแหน่งไปได้ตลอดหน่วยความจำสำรอง

ปกติตัวชี้ตำแหน่งจะอยู่ที่ต้นบรรทัดแรกหรืออยู่หลังบรรทัดสุดท้ายหรืออยู่ระหว่างตัวอักษรสองตัว ในกรณีที่ต้องการพิมพ์ข้อมูลหรือต้องการแก้ตัวอักษรที่ผิดเราต้องเลื่อนตัวชี้ตำแหน่งไปยังตำแหน่งตัวอักษรที่ผิด ตัวชี้จะเป็นตำแหน่งอ้างอิงของการใช้คำสั่งใน ED ดังนั้นจึงจำเป็นต้องรู้ตำแหน่งของตัวชี้ตำแหน่งตลอดเวลาในขณะที่พิมพ์ ตัวชี้ตำแหน่งจะชี้ตำแหน่งในหน่วยความจำตลอดเวลาและไม่สามารถเลื่อนไปไฟล์อื่นหรือไฟล์ชั่วคราวอื่นได้

การแทรกข้อความ

เมื่อเราเริ่มใช้ ED โปรแกรม หน่วยความจำสำรองจะว่างอยู่เราอาจอ่านข้อมูลจากไฟล์หลักเข้ามาในหน่วยความจำสำรองหรืออาจสร้างไฟล์ขึ้นมาใหม่หรืออาจแทรกข้อความโดยการใช้คำสั่ง I (Insert)

การใช้คำสั่ง I โดยการพิมพ์อักษร I หลังสัญลักษณ์ * ที่ ED พิมพ์ทางจอแล้วตามด้วยกดคีย์ Return (Carriage Return (< CR >)) ตัวชี้จะเลื่อนลงมา 1 บรรทัดและชี้ตอนต้นของหน่วยความจำสำรอง ดังตัวอย่าง

```
A > ED FILE.TYP  < CR >
*    I < CR >
```

เมื่ออยู่ภายใต้การทำงานของคำสั่ง I ED จะยอมรับบรรทัดทั้งหมดที่พิมพ์ลงไปและจะเก็บในหน่วยความจำสำรองเรียงตามลำดับ จนกว่าเมื่อเราจะบอก ED โปรแกรมว่าต้องการหยุดป้อนข้อมูลโดยการกดคีย์ CTRL และคีย์อักษร Z (CTRL-Z) พร้อมกัน ผลก็คือกลับสู่ ED

ตัวชี้ตำแหน่งจะยังคงอยู่ที่ตำแหน่งถัดจากตัวอักษรหลังสุดที่พิมพ์ (ขณะที่อยู่ภายใต้คำสั่ง I)เราอาจลบอักษรตัวสุดท้ายที่พิมพ์ได้โดยการกดคีย์ DELETE อักษรที่ถูกลบจะปรากฏซ้ำขึ้นบนจอเพื่อเป็นการบอกให้ผู้ใช้รู้ว่าอักษรตัวนี้ถูกลบออกไปแล้ว แต่ในบางเครื่องตัวสุดท้ายที่ลบนี้จะหายไป

ในกรณีที่พิมพ์ข้อความผิด ปรากฏอยู่ในบรรทัดเราอาจลบทั้งบรรทัดโดยการกด CTRL-U และจะปรากฏข้อความบรรทัดที่ลบปรากฏซ้ำบนจอเพื่อบอกให้ทราบว่าบรรทัดนี้ถูกลบแล้ว เมื่อต้องการออกจากคำสั่ง I ทำได้ด้วยการกด CTRL-Z

การออกจาก ED

สมมติขณะอยู่ใน ED เราต้องการสิ้นสุดการป้อนข้อมูลได้โดยการใช้คำสั่ง E หลังเครื่องหมาย "*"

คำสั่ง E จะเขียนข้อมูลจากหน่วยความจำสำรองลงบนไฟล์ชั่วคราว แล้วเปลี่ยนชื่อไฟล์ชั่วคราวเป็นชื่อไฟล์หลัก โดยชื่อของไฟล์หลักจะเปลี่ยนเป็น .BAK ไฟล์แล้วคืนการควบคุมให้ CP/M คำสั่งต่อไปนี้เป็นคำสั่งที่มีประโยชน์ในงานแต่ละอย่างระหว่างการพิมพ์ใน ED

H Reedit : ลักษณะการทำงานเหมือนกับทำตามคำสั่ง E ไฟล์ชั่วคราวเปลี่ยนเป็นไฟล์หลักไฟล์ใหม่ หน่วยความจำสำรองจะว่างและไฟล์ชั่วคราวไฟล์ใหม่จะได้รับการสร้างแล้ว ใช้คำสั่ง ED อีกครั้งโดยอัตโนมัติ

O origin : กลับสู่ไฟล์หลักเดิม หน่วยความจำสำรองจะลบว่างไว้ ไฟล์ชั่วคราวถูกลบแล้วไฟล์หลักจะถูกนำเข้ามาในหน่วยความจำสำรองใหม่ ตัวชี้ตำแหน่งจะชี้ที่ต้นหน่วยความจำสำรอง

Q Quit : ไม่มีไฟล์ไหนเปลี่ยนแปลงคืนการควบคุมแก่ CP/M

ข้อแนะนำ ใช้คำสั่ง H ระหว่างการพิมพ์ข้อความใน ED ถ้ามีการรบกวนเกิดขึ้น เช่น อาจจากไฟตก ส่วนของไฟล์ที่ถูกแก้ไขแล้วหลังจากใช้คำสั่ง H ครั้งหลังสุดจะยังคงเก็บอยู่บนไฟล์

การใช้คำสั่งควบคุมการพิมพ์ของ ED

ต่อไปนี้เป็นการอธิบายการใช้คำสั่งของ ED ที่เกี่ยวข้องกับการพิมพ์และหน่วยความจำสำรอง การทำงานของคำสั่งยังขึ้นอยู่กับตำแหน่งของตัวชี้ตำแหน่งอีกด้วย ซึ่งคำสั่งต่าง ๆ เหล่านี้แสดงอยู่ในตาราง และ n แทนค่าตัวเลขตั้งแต่ 0 ถึง 65535 ถ้าเว้นไว้จะถือว่ามีค่าเป็น 1 ดังที่ได้เคยกล่าวมาแล้วข้างต้น เช่น # อาจใช้แทน n หรือมีค่า 65535, + / - หมายถึงเครื่องหมาย + และ - และถ้าเว้นไว้จะสมมติว่ามีเครื่องหมายบวกอยู่ คำสั่งต่าง ๆ สามารถพิมพ์ต่อเนื่องกัน (และขึ้นอยู่กับหน่วยความจำที่ CP/M สำรองไว้ซึ่งสามารถจุได้ 256 ตัวอักษร) และจะทำงานต่อเมื่อกดคีย์ RETURN เท่านั้นและใน ED โปรแกรมสามารถใช้คำสั่งควบคุมการพิมพ์ของ CP/M ได้อีกด้วย เช่น DELETE, CTRL-U เพื่อลบบางอักษรของคำสั่งหรือลบคำสั่งทั้งบรรทัดออก เพื่อที่จะให้เกิดความเข้าใจเกี่ยวกับการใช้คำสั่งก็จะแสดงตัวอย่างการใช้ตามตารางที่ 6.1

ตัวอย่างการทำงานของคำสั่งควบคุมการพิมพ์ของ ED โปรแกรม

```
      แทนตัวชี้ตำแหน่ง

    * I<cr>
      NOW IS THE <cr>
      TIME FOR <cr>
      ALL GOOD MEN <cr>
      CTRL-Z
```

ตารางที่ 6.1 คำสั่งพื้นฐานที่ใช้ในการพิมพ์

ความหมาย	คำสั่ง	คำอธิบาย
เลื่อนไปต้นหรือท้าย	+ / – B	ถ้าใช้ + หมายถึงเลื่อนตัวชี้ตำแหน่งไปต้นหน่วยความจำสำรอง ถ้าใช้ - หมายถึงเลื่อนตัวชี้ตำแหน่งไปตอนท้ายหน่วยความจำสำรอง
เลื่อนผ่านตัวอักษร	+ / – nC	เลื่อนตัวชี้ตำแหน่งไปทางต้นหน่วยความจำสำรองถ้าเป็น + และไปทางท้ายหน่วยความจำสำรองถ้าเป็น - โดยการเลื่อนใช้นับตัวอักษรและนับ<CR><IF> เป็นตัวอักษร 2 ตัว
ลบอักษร	+ / – nD	ถ้าใช้ +nD จะลบอักษร n ตัวไปทางขวา ถ้า - ก็จะลบมาทางซ้าย
ลบบรรทัด	+ / – nK	ถ้าใช้ +nK จะลบ n บรรทัดจากตำแหน่งที่ตัวชี้ตำแหน่งอยู่ ถ้าตัวชี้ตำแหน่งไม่ได้อยู่ต้นบรรทัด ตัวอักษรที่อยู่ทางซ้ายจะไม่ได้รับการลบ
เลื่อนบรรทัด	+ / – nL	ถ้า n = 0 ตัวชี้ตำแหน่งจะเลื่อนไปอยู่ต้นบรรทัด ถ้า n ≠ 0 ตัวชี้ตำแหน่งจะเลื่อนไป n บรรทัดการเลื่อนจะเลื่อนขึ้นถ้าเป็น - และจะเลื่อนลงถ้าเป็น +
แสดงบรรทัดบนจอ	+ / – nT	ถ้า n = 0 จะแสดงข้อความบรรทัดที่ตัวชี้ตำแหน่งอยู่บนจอจากตำแหน่งตัวชี้ตำแหน่งไปทางซ้ายจนถึงต้นบรรทัด ถ้า n = 1 จะแสดงข้อความจากตัวชี้ตำแหน่งไปทางขวาจนสุดบรรทัด ถ้า n>1 หรือเป็นบวก ผลจะให้พิมพ์ n บรรทัด จากตำแหน่งที่ตัวชี้ตำแหน่ง ในทำนองเดียวกันถ้าใช้ - จะพิมพ์ n บรรทัดขึ้นไปข้างบนจากตำแหน่งตัวชี้ตำแหน่งอยู่
เลื่อนไปบรรทัดที่ระบุแล้วพิมพ์ข้อความบรรทัดนั้น	+ / – n	เหมือนกับ + / – nLT ซึ่งเคลื่อนขึ้นลงตามเครื่องหมาย -, +

อธิบายเราตอบรับ * ด้วยคำสั่ง I บรรทัดเลื่อนลงมา 1 บรรทัด เริ่มพิมพ์ข้อความเมื่อจบข้อความกดคีย์ Return (<cr>) (ซึ่ง <Lf> จะถูกนำมาต่อท้ายโดยอัตโนมัติ ซึ่ง <lf> เป็นคำสั่งเลื่อนบรรทัดแต่ <cr> แสดงว่าความยาวของบรรทัดมีแค่นี้)

เมื่อจบโดย CTRL-Z แล้ว ข้อมูลที่เก็บในหน่วยความจำสำรอง เก็บในรูปของรหัส ASCII โดยมีรหัส ASCII ของ <cr> และ <lf> ต่อท้าย

NOW IS THE<cr><lf>
TIME FOR <cr><Lf>
ALL GOOD MEN <cr><lf> △

กลุ่มคำสั่ง	ผลบนจอ	ตำแหน่งการชี้ของตัวชี้ตำแหน่งในหน่วยความจำ
B2T<cr>	NOW IS THE	△ NOW IS THE <cr>
	TIME FOR	TIME FOR <cr><lf>
		ALL GOOD MEN<cr><lf>
5COT<cr>	NOW I	NOW I △S THE<cr><lf>
		TIME FOR<cr><lf>
		ALL GOOD MEN<cr><lf>
2L-T	TIME FOR	NOW IS THE<cr><lf>
		TIME FOR<cr><lf>
		△ ALL GOOD MEN <cr><lf>
– L # K<cr>		NOW IS THE<cr><lf> △
I<cr>		NOW IS THE<cr><lf>
TIME TO<cr>		TIME TO<cr><lf>
INSERT<cr>		INSERT<cr><lf> △
CTRL-Z		
2L- #T<cr>	NOW IS THE	NOW IS THE<cr><lf>
		△ TIME TO<cr><lf>
		INSERT<cr><lf>
B # T	NOW IS THE	NOW IS THE<cr><lf>
	TIME TO	TIME TO<cr><lf>
	INSERT	INSERT<cr><lf>
FM		NOW IS THE<cr><lf>
		TIM △ E TO<cr><lf>
		INSERT<cr><lf>
BFS CTRL-Z-2DOLI	THIS IS CTRL-Z	THIS IS NOW THE
		<cr><lf>
		TIME TO<cr><Lf>
		INSERT<cr><lf>

หมายเหตุ ตัวชี้ตำแหน่งเพียงแต่ชี้ไปยังตัวตำแหน่งต่าง ๆ ในหน่วยความจำสำรองเท่านั้นไม่ได้เข้าอยู่ในหน่วยความจำสำรองด้วย

การเรียงของข้อมูลในหน่วยความจำสำรองเรียงต่อเนื่องกันไปแต่คอมพิวเตอร์สามารถแยกบรรทัดได้โดยรหัส ASCII ของ<cr>และ<lf>

เมื่อมีการแทรกข้อมูลที่อยู่ด้านหลังตัวชี้ตำแหน่งจะถูกเลื่อนออกไปเพื่อให้มีที่ว่างและการลบตัวอักษรทำให้ข้อความทางขวาเลื่อนเข้ามาแทนที่ที่ว่างนั้น

การใช้คำสั่งค้นหากลุ่มข้อความและเปลี่ยนแปลงข้อความ

ในบางครั้งเราอาจต้องการหาข้อความบางกลุ่มในไฟล์และต้องการจะแก้ไขตัวอย่างเช่นมีคำว่า Model-8080 อยู่หลายคำในไฟล์ซึ่งเราต้องการเปลี่ยนเป็น Model-6800 แทนหรืออาจต้องการเปลี่ยนแปลงคำว่า DELAY ที่อยู่ในไฟล์ด้วยคำว่า TIME DELAY

คำสั่งการค้นหา

n F<text>

คำสั่ง F มีความหมายว่าให้ค้นหากลุ่มคำ<text>ตัวที่ n ที่ต้องการให้ปรากฏบนจอและถ้าต้องการเราอาจมีคำสั่งอื่นต่อท้ายอีกในบรรทัดเดียวกัน ED โปรแกรมจะไม่ทำตามคำสั่งจนกว่าจะกดคีย์ RETURN

ED โปรแกรมจะค้นหากลุ่มคำไปข้างหน้าจากตำแหน่งที่ตัวชี้ตำแหน่งอยู่เมื่อหาเจอตัวชี้ตำแหน่งจะอยู่ที่หลังอักษรตัวสุดท้ายของกลุ่มคำที่ค้นเจอและในกรณีไม่มีข้อความที่ต้องการหา ตัวชี้ตำแหน่งจะชี้อยู่ที่เดิมและถ้า n เว้นไว้จะถือว่ามีค่าเป็น 1

เมื่อ ED โปรแกรมค้นหากลุ่มคำที่ต้องการไม่พบจะพิมพ์ # บนจอและไม่ทำตามคำสั่งที่ตามมาในบรรทัดเดียวกัน

ปกติมีคำสั่งที่ทำงานคล้ายกับคำสั่งค้นหา F แต่สามารถอ่านเพิ่มเติมจากไฟล์หลังและเขียนลงในไฟล์สำรองโดยอัตโนมัติดังรูปแบบคำสั่งข้างล่างนี้

n N<text>

ซึ่ง n มีความหมายว่าค้นหาเอกสาร (text)กลุ่มที่ n จากเอกสารแต่จากคำสั่ง F ตรงที่ว่าเมื่อหาไม่พบในหน่วยความจำสำรองข้อความทั้งหมดในหน่วยความจำสำรองจะได้รับการเขียนกลับลงไปบนดิสค์ (เหมือนกับการทำงานของคำสั่ง # W) และจะมีการอ่านข้อมูลจากไฟล์หลักจนกระทั่งหน่วยความจำสำรองจุได้ครึ่งหนึ่งหรือไฟล์หลักที่เหลือทั้งหมดถูกอ่านเข้ามาหมดและการค้นหาจะดำเนินต่อไปจนกระทั่งกลุ่มตัวอักษร<text> กลุ่มที่ n ถูกค้นพบหรือจนกระทั่งไฟล์หลักถูกย้ายไปเก็บในไฟล์สำรองจนหมด

การแทรกข้อมูลเราใช้คำสั่ง I ซึ่งมักใช้ร่วมกับคำสั่ง

I<text>

เมื่อ <text>คือกลุ่มตัวอักษรที่ต้องการแทรกเข้าไปและการแทรกกลุ่มตัวอักษรต้องสิ้นสุดลงด้วย CTRL-Z ปิดท้าย การแทรกจะแทรกหลังตัวชี้ตำแหน่งเมื่อเสร็จตัวชี้ตำแหน่งจะอยู่ที่หลังตัวอักษรตัวสุดท้ายที่แทรก

ED มีคำสั่งคล้ายกับคำสั่ง F และคำสั่ง I ทำงานร่วมกันเรียกว่าคำสั่งการแทนที่ซึ่งมีรูปแบบการใช้คำสั่งดังนี้

n S <old text> CTRL-Z <new text>

คำสั่ง S หมายถึงการค้นหาแล้วแทนที่ ดังนั้นกลุ่มคำ<old text>จะถูกแทนโดยกลุ่มคำ<New text>โดยจะทำการค้นหาแล้วแทนที่ n ครั้งแล้วตัวชี้ตำแหน่งจะอยู่หลังตัวอักษรตัวสุดท้ายของ<new text>ถ้า<old text> หาไม่พบ # จะถูกแสดงบนจอและรอคำสั่งต่อไป

ไฟล์ไลบรารี่ (libraries file)

มีบ่อยครั้งที่เราอยากที่จะให้โปรแกรมของเรามีหมายเหตุหรือข้อความบอกลิขสิทธิ์ดังนั้นข้อความที่เราต้องการใช้บ่อย ๆ นี้สามารถเก็บไว้ในไฟล์ที่เรียกว่า ไลบรารี่ไฟล์ อย่างไรก็ตามไม่มีข้อความในไลบรารี่ไฟล์โดยจนกว่าเราจะเขียนมันลงไป โดยการใช้ ED โปรแกรมโดยกำหนดให้มีชนิดของไฟล์เป็น .LIB

ED โปรแกรมอนุญาตให้สามารถรวมไฟล์ที่แยกจากกันเข้าด้วยกันในระหว่างการใช้ ED โปรแกรมโดยการใช้คำสั่ง R

R<file name>

คำสั่ง R หมายถึงอ่านไฟล์บนดิสค์ซึ่งมีชนิดของไฟล์ .LIB และวางลงในหน่วยความจำสำรองหลังตัวชี้ตำแหน่ง

คำสั่งกำหนดให้ทำซ้ำ

nM <กลุ่มคำสั่ง>

โดย ED จะทำตามกลุ่มคำสั่งตามลำดับ n ครั้งและถ้า n เว้นไว้หรือมีค่าเป็น 0 หรือ 1 กลุ่มคำสั่งจะทำไปเรื่อย ๆ จนกว่าจะมีข้อผิดพลาดเกิดขึ้น (เช่น ถึงจุดสุดท้ายของหน่วยความจำสำรองซึ่งเกิดจากคำสั่ง F)

ถ้ากลุ่มคำสั่งมีคำสั่ง I และ S รวมอยู่ด้วย คำสั่งเหล่านี้ต้องปิดท้ายด้วย CTRL-Z เหมือนกับ <cr>(Carriage return) บอกการสิ้นสุดของบรรทัด ตัวอย่าง

10M I REPEAT ^Z<cr>

ความหมายแทรกกลุ่มคำ REPEAT ลงไป 10 ครั้ง

อักษรตัวใหญ่และอักษรตัวเล็ก

ในการใช้คำสั่ง I, F คำสั่งจะเปลี่ยนตัวอักษรใน<text>เป็นอักษรตัวใหญ่ถ้า I, F, etc. พิมพ์ด้วยอักษรตัวใหญ่ ถ้าคำสั่งพิมพ์เป็นอักษรตัวเล็ก (i, fเป็นต้น) จะไม่มีการเปลี่ยนเกิดขึ้น

คำสั่ง U.

อาจใช้ใน ED เพื่อเปลี่ยนตัวอักษรเป็นตัวพิมพ์ใหญ่แล้วนำไปเก็บในหน่วยความจำสำรองแต่ที่ปรากฏบนจอก็เป็นไปตามการพิมพ์ไม่มีการเปลี่ยนแปลงและคำสั่ง

-U

เป็นการหยุดการเปลี่ยนตัวอักษรตัวเล็กเป็นอักษรตัวใหญ่

การแสดงสัญลักษณ์ของความผิดพลาด

ในการใช้คำสั่งใน ED อาจมีการใช้ผิดพลาดทำให้การทำงานผิดไม่สามารถเป็นไปได้ เช่น การค้นหาอักษรที่ไม่มีอยู่ในไฟล์หรือพิมพ์คำสั่งด้วยอักษรที่ไม่ใช่คำสั่ง ED โปรแกรมจะแสดงสัญลักษณ์

เพื่อบอกลักษณะของความผิดพลาดออกทางจอ

การค้นหาล้มเหลวเกิดจากการใช้คำสั่ง F, S และ N

??X ตัวอักษร X ไม่ใช่เป็นคำสั่ง หรืออาจเป็นคำสั่ง E, H, Q หรือ Q ซึ่ง Q ไม่สามารถใช้ประกอบเป็นกลุ่มคำสั่งได้

00 LIB ไฟล์ไม่สามารถหาพบ เกิดจากการใช้คำสั่ง R

>> หน่วยความสำรองเต็ม (อาจใช้คำสั่ง D, K, N, S หรือ W เพื่อลดขนาดไฟล์ลง) หรือ F, N, S เปลี่ยนกลุ่มอักษรที่มีขนาดยาวเกินไป

ถ้าหน่วยความจำสำรองเต็มขณะใช้คำสั่งA ED จะแสดงสัญลักษณ์ ≫A ทางจอและตัวชี้ตำแหน่งจะอยู่ที่ตำแหน่งสุดท้ายของหน่วยความจำสำรอง (ซึ่งตามปกติ ตัวชี้ตัวแหน่งจะไม่เปลี่ยนตำแหน่งเนื่องจากใช้คำสั่ง A)

ถ้าดิสค์เต็มหรือสารบัญ (directory) เต็มเมื่อ ED ต้องการสร้างหรือขยายไฟล์หรือข้อผิดพลาดอย่างอื่นเกิดขึ้นกับไฟล์ในดิสค์จะปรากฏข้อความบนจอ

FILE ERROR

และการควบคุมจะกลับไปยัง CP/M

ถ้าการพิมพ์ล้มเหลว (อาจเนื่องจาก FILE ERROR, ^C (Warm boot), คำสั่ง Q, ไฟตก ฯลฯ) ไฟล์หลักจะไม่เปลี่ยน BAK ไฟล์อันก่อนจะถูกลบและไฟล์ชั่วคราว ($$$) จะถูกปล่อยให้ว่างไว้หรือไม่สมบูรณ์

ดังนั้นเมื่อการพิมพ์เกิดขึ้นโดยสมบูรณ์แล้วแต่ภายหลังพบว่ามันมีข้อผิดพลาด (อาจเกิดจากคำสั่ง H ถ้าคำสั่ง H ถูกใช้) สามารถแก้ไขได้โดยการลบไฟล์อันใหม่นี้แล้วทำการเปลี่ยนชื่อไฟล์ BAK เป็นไฟล์เดิม เราอาจตรวจรอบ ข้อมูลใน BAK ไฟล์ก่อนโดยใช้คำสั่ง TYPE

TYPE FILE.BAK

เมื่อถูกต้องใช้คำสั่ง ERA

ERA FILE.TXT

แล้วทำการเปลี่ยนชื่อ BAK ไฟล์

REN FILE.TXT = FILE.TXT

แล้วทำการใช้ ED โปรแกรมแก้ไขใหม่ต่อไป

มีเอดิเตอร์โปรแกรมมากมายในท้องตลาดซึ่งสามารถใช้กับ CP/M จากการนับครั้งหลังสุดมีถึงมากกว่า 50 โปรแกรม ในวันหนึ่งคุณอาจต้องใช้เอดิเตอร์ใหม่เหล่านี้ ในการเรียนเอดิเตอร์ใหม่อาจเรียนได้เร็วถ้าเราวางแผนผังของเอดิเตอร์โปรแกรมนั้นลงข้าง ๆ แป้นพิมพ์เพราะเอดิเตอร์เหล่านี้มักใช้แป้นพิมพ์ในการควบคุมการทำงาน ดังนั้นจึงไม่จำเป็นที่เราต้องกลัวที่จะเรียนรู้เอดิเตอร์ใหม่ ๆ เหล่านี้เลย

นอกจากนี้ยังมีโปรแกรมสร้างเอกสารจำพวกเวิร์ดโปรเซสซิ่งอีกหลายตัวซึ่งมีขีดความสามารถสูงมาก หากมีให้เลือกใช้แล้ว ขอแนะนำให้ใช้โปรแกรมเวิร์ดโปรเซสซิ่งเหล่านั้น และในบรรดาเวิร์ดโปรเซสซิ่งที่มีขีดความสามารถสูงตัวหนึ่งคือ เวิร์ดสตาร์ (Word star)

```
A>
A>
A>                         → เรียกโปรแกรมโดยการใช้ ED และตามด้วยชื่อและนามสกุลของไฟล์
A>
A>ED SAMPLE.TXT
                   → ไฟล์ใหม่
NEW FILE           → คำสั่งในการแทรกข้อความลงในหน่วยความจำบัฟเฟอร์
   : *VI
  1:    ED IS THE NAME OF THE PARTICULAR EDITOR THAT
  2:  COME WITH THE CP/M OPERATING SYSTEM. ED IS A "LINE ORIENTED"
  3:  TEXT EDITOR , AND NOT A WORD PROCESSOR IN THE USUAL SENSE.
  4:
   : *B3T    → คำสั่งให้เอดิเตอร์พิมพ์ข้อความจากบัฟเฟอร์ขึ้นมาแสดงบนจอภาพ ตั้งแต่บรรทัดแรกนับมา 3 บรรทัด
  1:    ED IS THE NAME OF THE PARTICULAR EDITOR THAT
  2:  COME WITH THE CP/M OPERATING SYSTEM. ED IS A "LINE ORIENTED"
  3:  TEXT EDITOR , AND NOT A WORD PROCESSOR IN THE USUAL SENSE.
  1: *
  1: *1:STHAT^ZTHAT WILL → คำสั่งค้นหาคำว่า THAT ตั้งแต่บรรทัดที่ 1 ถ้าเจอแล้วจะแทนด้วยคำว่า THAT WILL
  1: *B3T
  1:    ED IS THE NAME OF THE PARTICULAR EDITOR THAT WILL
  2:  COME WITH THE CP/M OPERATING SYSTEM. ED IS A "LINE ORIENTED"
  3:  TEXT EDITOR , AND NOT A WORD PROCESSOR IN THE USUAL SENSE.
  1: *E
        → คำสั่งในการเก็บข้อความในบัฟเฟอร์ลงในดิสค์/สิ้นสุดการแก้ไข
```

รูปที่ 6.2 ตัวอย่างการใช้งานโปรแกรมเอดิเตอร์ (ED)

ตัวอย่างการเขียนโปรแกรมภาษาแอสเซมบลี้ 8080

สืบเนื่องจากโปรแกรมเอดิเตอร์ทำหน้าที่เป็นเสมือนตัวช่วยเขียนโปรแกรมในภาษาต่าง ๆ ก่อนที่จะนำไปใช้ในการคอมพลายด์หรือให้คอมพิวเตอร์ทำงาน เอดิเตอร์ ED ที่อยู่ในโปรแกรมจัดระบบงาน CP/M ภาษาแอสเซมบลี้เป็นอีกภาษาหนึ่งที่สามารถใช้งานได้ดีโดยสามารถใช้ภาษาแอสเซมบลี้นี้เข้าไปดัดแปลงแก้ไขระบบซอฟต์แวร์หรือสร้างซอฟต์แวร์ต่าง ๆ ได้

ตัวอย่างที่จะใช้เอดิเตอร์เขียนนี้เป็นโปรแกรมภาษาแอสเซมบลี้ที่จะเขียนนี้ เป็นโปรแกรมย่อยเพื่อกำหนดเป็นการหน่วงเวลาและกำหนดเวลาอย่างละเอียดเพื่อใช้เป็นนาฬิกา โปรแกรมย่อยนี้เขียนขึ้นเพื่อใช้กับซีพียู 8085 ที่ทำงานด้วยสัญญาณนาฬิกา 3MHz โดยการกำหนดเวลาด้วยการลูปการทำงาน

การเริ่มใช้ ED ด้วยการกำหนดชื่อไฟล์ ASM ว่า FILE ASM.ASM EDจะตอบสนองด้วยคำว่า NEWFILE จากนั้นผู้ใช้จะตอบกลับด้วยการใช้ V หมายถึง verity คือการใส่หมายเลขกำกับหน้าบรรทัดเพื่ออ้างถึงระหว่างการทำงาน ส่วน I คือการจะป้อนข้อมูลเข้า

```
ED FILEASM.ASM

NEW FILE
   : *VI
  1:   ;PROGRAM FOR DEMONSTRATION
  2:   ;DATE:      MARCH 1,1984
```

```
  3:  ;
  4:            ORG      8000H    ;ORIGIN PROGRAM AT 8000H
  5:  OUTPUT:   EQU      0A1H     ;OUTPUT PORT NUMBER
  6:  ON:       EQU      01H      ;TURN ON BIT 0
  7:  LTIME:    EQU      100      ;1MS LOOP COUNT
  8:  ;
  9:  START:    JMP      MAIN
 10:  COUNT:    DB       100      ;SET COUNT AT THIS LOCATION (100MS)
 11:  ;
 12:  MAIN:     PUSH     H        ;SAVE ALL REGISTERS
 13:            PUSH     D
 14:            PUSH     B
 15:            PUSH     PSW
 16:  ;
 17:            MVI      A,0      ;CLEAR ACCUMULATOR
 18:            OUT      OUTPUT   ;SET BIT 0 TO 0
 19:            LDA      COUNT    ;GET X MS COUNT
 20:            MOV      C,A      ;SAVE IT
 21:            MVI      B,LTIME  ;LOOP COUNT = 1 MS
 22:            MVI      A,ON     ;SET BIT 0 TO 1
 23:            OUT      OUTPUT   ;START PERIOD
 24:  ;
 25:  LOOPT:    DCR      B        ;DECREMENT LOOP COUNT
 26:            NOP               ;DELAY
 27:            NOP
 28:            NOP
 29:            NOP
 30:            JNZ      LOOPT    ;LOOP IF NOT OK
 31:  ;
 32:            MVI      B,LTIME  ;RESET LOOP COUNT
 33:            DCR      C        ;DECREMENT 1 MS
 34:            JNZ      LOOPT    ;GO AGAIN
 35:  ;
 36:            MVI      A,0      ;CLEAR ACCUMULATOR
 37:            OUT      OUTPUT   ;END PERIOD
 38:  ;
 39:            POP      PSW      ;RESTORE REGISTER
 40:            POP      B
 41:            POP      D
 42:            POP      H
 43:  ;
 44:            RET               ;RETURN TO MAIN PROGRAM
 45:            END
 46:
   : *45T
   : *B45T
  1:  ;PROGRAM FOR DEMONSTRATION
  2:  ;DATE:    MARCH 1,1984
  3:  ;
  4:            ORG      8000H    ;ORIGIN PROGRAM AT 8000H
  5:  OUTPUT:   EQU      0A1H     ;OUTPUT PORT NUMBER
  6:  ON:       EQU      01H      ;TURN ON BIT 0
  7:  LTIME:    EQU      100      ;1MS LOOP COUNT
  8:  ;
  9:  START:    JMP      MAIN
 10:  COUNT:    DB       100      ;SET COUNT AT THIS LOCATION (100MS)
 11:  ;
 12:  MAIN:     PUSH     H        ;SAVE ALL REGISTERS
 13:            PUSH     D
```

```
                PUSH    B
                PUSH    PSW
        ;
                MVI     A,0      ;CLEAR ACCUMULATOR
                OUT     OUTPUT   ;SET BIT 0 TO 0
                LDA     COUNT    ;GET X MS COUNT
                MOV     C,A      ;SAVE IT
                MVI     B,LTIME  ;LOOP COUNT = 1 MS
                MVI     A,ON     ;SET BIT 0 TO 1
                OUT     OUTPUT   ;START PERIOD
                ;
        LOOPT:  DCR     B        ;DECREMENT LOOP COUNT
                NOP              ;DELAY
                NOP
                NOP
                NOP
                JNZ     LOOPT    ;LOOP IF NOT OK
        ;
                MVI     B,LTIME  ;RESET LOOP COUNT
                DCR     C        ;DECREMENT 1 MS
                JNZ     LOOPT    ;GO AGAIN
        ;
                MVI     A,0      ;CLEAR ACCUMULATOR
                OUT     OUTPUT   ;END PERIOD
        ;
                POP     PSW      ;RESTORE REGISTER
                POP     B
                POP     D
                POP     H
        ;
                RET              ;RETURN TO MAIN PROGRAM
                END
        *E
```

7

ASM แอสเซมเบลอร์ของ CP/M

โดยปกติภาษาเบสิค ฟอร์แทรน โคบอล และภาษาระดับสูงอื่น ๆ จะเหมาะสำหรับการใช้งานโดยทั่วไปเพราะมีความสัมพันธ์ของรูปแบบคำสั่ง (key word) ที่ง่ายต่อความเข้าใจและมีคำสั่งที่ใช้ในการทำงานที่มีลักษณะการทำงานซ้ำกันอย่างต่อเนื่องหรือทำงานตามจำนวนครั้งที่กำหนด

สำหรับภาษาเบสิคการทำงานวนรอบ (loop) ใช้คำสั่ง FOR......NEXT ดังตัวอย่าง

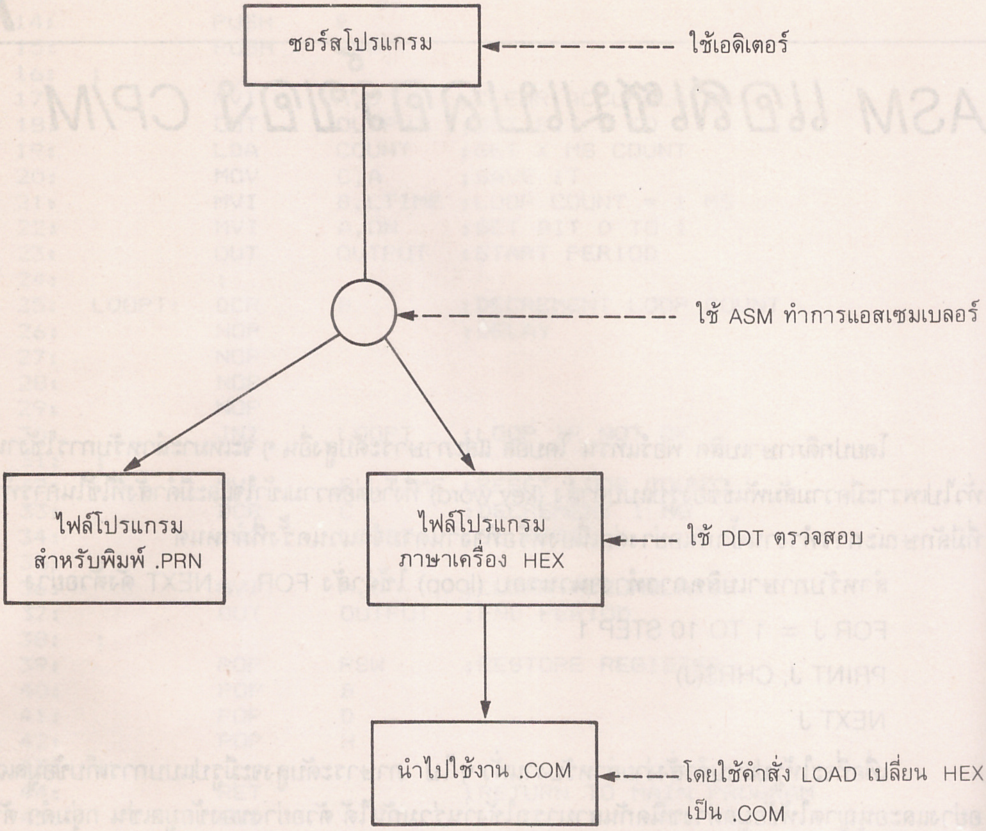
```
FOR J = 1 TO 10 STEP 1
PRINT J, CHR$(J)
NEXT J
```

เพื่อที่จะให้รูปแบบคำสั่งง่ายสำหรับงานทั่ว ๆ ไป ภาษาระดับสูงจะมีรูปแบบการเก็บข้อมูลเฉพาะอย่างและอนุญาตให้ข้อมูลต่างชนิดกันสามารถใช้งานร่วมกันได้ ตัวอย่างของข้อมูลเช่น กลุ่มคำ ตัวเลข เงื่อนไขจริง (True) และเงื่อนไขไม่จริง (Fault) เป็นต้น สำหรับภาษาฟอร์แทรนจะมีรูปแบบของการเก็บของข้อมูลตัวเลขที่ดีมาก เช่นมีการทำงานเกี่ยวกับเมตริกซ์หลายแบบจึงทำให้เหมาะสมสำหรับงานทางด้านวิศวกรรมศาสตร์และงานวิจัยทางด้านวิทยาศาสตร์

ภาษาโคบอลมีลักษณะโครงสร้างของภาษาแบบไฟล์จึงเหมาะสมกับงานทางด้านธุรกิจ อย่างไรก็ตามภาษาระดับสูงไม่ใช่การแก้ปัญหาที่ดีที่สุดสำหรับงานทุกอย่าง งานบางอย่างที่ภาษาระดับสูงไม่สามารถทำได้หรือทำได้ไม่ดี ซึ่งในสถานการณ์เช่นนี้โปรแกรมแอสเซมบลี้อาจทำได้หรือทำได้ดีกว่าแต่ก็ยากต่อการเรียนรู้มากกว่าภาษาระดับสูง

ในการทำงานของเครื่องคอมพิวเตอร์นั้นโปรแกรมภาษาระดับสูงต้องเปลี่ยนรูปแบบคำสั่ง (Syntex) และโครงสร้างของภาษาระดับสูง ไปเป็นรูปแบบคำสั่งและโครงสร้างของภาษาเครื่องซึ่ง CPU สามารถเข้าใจและทำงานได้

ตัวแปลภาษาเบสิคแบบอินเตอร์พรีทเตอร์ (interpretor) จะเปลี่ยนประโยคภาษาเบสิคเป็นประโยค (Statement) ภาษาเครื่องซึ่งคำสั่งภาษาเครื่องที่แปลนี้จะมีจำนวนคำสั่งมากกว่าจำนวนคำสั่งในภาษาเบสิค การแปลภาษาเบสิคเป็นภาษาเครื่องนั้นจะได้โปรแกรมภาษาเครื่องหลายพันคำสั่ง ดังนั้น โปรแกรมภาษาเบสิคจะทำงานช้ากว่าโปรแกรมภาษาเครื่องที่เขียนโดยตรงที่มีการทำงานอย่างเดียวกัน จะเห็นได้ว่าความเร็วเป็นอีกสาเหตุหนึ่งที่ทำให้ภาษาแอนเซมบลี้ได้รับการเลือกมาใช้ในงานบางอย่าง



รูปที่ 7.1 โครงสร้างการพัฒนาโปรแกรมด้วยภาษาแอสเซมบลี้

แอสเซมเบลอร์ เป็นโปรแกรมใช้งานตัวหนึ่งที่จะคอมพลายตัวโปรแกรมภาษาแอสเซมบลี้จึงช่วยทำให้ผู้ใช้เขียนโปรแกรมซึ่งเครื่องคอมพิวเตอร์สามารถอ่านและทำงานได้โดยตรงโดยไม่ต้องเขียนเป็นรหัสภาษาเครื่อง

ASM เป็นแอสเซมเบลอร์ตัวหนึ่งใช้ในการสร้างรหัสภาษาเครื่องของไมโครโปรเซสเซอร์ 8080, 8085 จากภาษาแอสเซมบลี้ของไมโครโปรเซสเซอร์เหล่านั้น

แอสเซมเบลอร์เปรียบเสมือนตัวเปลี่ยนซึ่งเปลี่ยนจากสิ่งหนึ่งไปเป็นอีกสิ่งหนึ่งเหมือนกับการทำงานของฟังค์ชัน (เช่น $f(x) = x^2 + 2x + 3$) แต่แตกต่างกันในรูปแบบการทำงาน ในเครื่องคอมพิวเตอร์แอสเซมเบลอร์จะเปลี่ยนสัญลักษณ์คำสั่ง (mnemonic) และกลุ่มตัวอักษรต่อเนื่องที่ใช้กำหนดบรรทัด (label) ที่เขียนโดยใช้ ED เป็นกลุ่มของตัวเลขที่ต่อเนื่องที่เครื่องรู้จัก

ตัวเลขที่ใช้เป็นคำสั่งเป็นสิ่งที่เครื่องคอมพิวเตอร์สามารถเข้าใจได้ กลุ่มตัวเลขต่อเนื่องที่แปลโดยแอสเซมเบลอร์เรียกว่า *อ็อบเจ็กต์โค้ด (object code)* หรือรหัสภาษาเครื่องแต่รหัสสัญลักษณ์คำสั่งจะเรียกว่า *แอสเซมบลี้โค้ด* แอสเซมบลี้โค้ดง่ายต่อการเข้าใจมากกว่าอ็อบเจ็กต์โค้ด เช่น 00 EA 4C C3 60 00 เป็นอ็อบเจ็กต์โค้ด (แสดงโดยเลขฐาน 16) จะเขียนแทนด้วยภาษาแอสเซมบลี้ได้ดังนี้ NOP LDA JMP 0060 จะเห็นว่าคำสั่งภาษาแอสเซมบลี้มีลักษณะง่ายต่อความเข้าใจมากกว่า

สำหรับบางคนจะเห็นภาษาแอสเซมบลี้ยากต่อการเรียนรู้ดังนั้นเพื่อที่จะลดความสับสนและง่ายต่อความเข้าใจยิ่งขึ้น อาจจะเขียนโปรแกรมภาษาแอสเซมบลี้สั้น ๆ แล้วใช้แอสเซมเบลอร์เปลี่ยนเป็นภาษาเครื่อง เพื่อที่จะเข้าใจการทำงานของแอสเซมเบลอร์

โปรแกรมต่อไปนี้จะทำงานโดยการสร้างพัลส์ที่มีเวลาแน่นอนเป็นวินาทีที่บิทสูงสุด (bits) คือบิท D7 ของ พอร์ท F0 โปรแกรมนี้อาจเรียกใช้โดยภาษาเบสิค ทั้งนี้เนื่องจากโปรแกรมภาษาเบสิคไม่สามารถกำหนดเวลาเริ่มต้นและเวลาหยุดช่วงของพัลส์ได้แน่นอนเพียงพอ โปรแกรมนี้เขียนขึ้นเพื่อให้ทำงานกับ 8085 ไมโครโปรเซสเซอร์โดยใช้สัญญาณนาฬิกา 3 MHz ความเร็วของสัญญาณนาฬิกามีความสำคัญต่อการทำงานในเรื่องลูป (loop) (เริ่มต้นด้วยคำสั่ง DCR B ไปถึง JNZ Loop) ไมโครโปรเซสเซอร์อาจใช้สัญญาณนาฬิกาต่างกัน ดังนั้นเราอาจเพิ่มหรือลด NOP เพื่อให้การทำงานวนรอบใช้เวลา 1 วินาที

```
TYPE FILEASM.ASM
;PROGRAM FOR DEMONSTRATION
;DATE:    MARCH 1,1984
;
         ORG      8000H     ;ORIGIN PROGRAM AT 8000H
OUTPUT:  EQU      0A1H      ;OUTPUT PORT NUMBER
ON:      EQU      01H       ;TURN ON BIT 0
LTIME:   EQU      100       ;1MS LOOP COUNT
;
START:   JMP      MAIN
COUNT:   DB       100       ;SET COUNT AT THIS LOCATION
                            (100MS)
;
MAIN:    PUSH     H         ;SAVE ALL REGISTERS
         PUSH     D
         PUSH     B
         PUSH     PSW
         MVI      A,0       ;CLEAR ACCUMULATOR
         OUT      OUTPUT    ;SET BIT 0 TO 0
         LDA      COUNT     ;GET X MS COUNT
         MOV      C,A       ;SAVE IT
         MVI      B,LTIME   ;LOOP COUNT = 1 MS
         MVI      A,ON      ;SET BIT 0 TO 1
         OUT      OUTPUT    ;START PERIOD
         ;
LOOPT:   DCR      B         ;DECREMENT LOOP COUNT
         NOP                ;DELAY
         NOP
         NOP
         NOP
         JNZ      LOOPT     ;LOOP IF NOT OK
         MVI      B,LTIME   ;RESET LOOP COUNT
         DCR      C         ;DECREMENT 1 MS
```

```
        JNZ         LOOPT     ;GO AGAIN
;
        MVI         A,0       ;CLEAR ACCUMULATOR
        OUT         OUTPUT    ;END PERIOD
;
        POP         PSW       ;RESTORE REGISTER
        POP         B
        POP         D
        POP         H
;
        RET                   ;RETURN TO MAIN PROGRAM
        END
```

โปรแกรมภาษาเบสิคอาจใช้คำสั่ง POKE (หรือคำสั่งที่ทำงานติดต่อกับหน่วยความจำได้เหมือนกัน) ซึ่งใช้ในการนำข้อมูลไปไว้ที่หน่วยความจำ (RAM) โดยตรงเพื่อส่งผ่านตัวเลขที่กำหนดช่วงความยาวของเวลาเป็นวินาทีของพัลซ์การเรียกใช้โปรแกรมจะ POKE ตัวเลขไปยังหน่วยความจำตำแหน่ง D803 และเริ่มทำงานที่ D800 โดยคำสั่ง CALL เพื่อเริ่มต้นสร้างพัลซ์

ตัวโปรแกรมนี้จะกระโดดข้ามตำแหน่งเก็บจำนวนการนับวินาทีของหน่วยความจำไปตำแหน่งเริ่มต้นของโปรแกรมหลัก รีจีสเตอร์จะถูกเก็บรักษาไว้ในแสตค (Stack) โดยใช้คำสั่ง PUSH เราต้องการกำหนดบิท 7 ของเอาต์พุต พอร์ทเป็น 0 โดยการเคลียร์รีจีสเตอร์ A เป็น 0 แล้วส่งข้อมูลในรีจีสเตอร์ A ไปยังพอร์ทแล้วนำตัวเลขจำนวนวินาทีที่ต้องการนับ (ที่ส่งมาจากโปรแกรมภาษาเบสิค) จากหน่วยความจำตำแหน่ง D803H แล้วเก็บในรีจีสเตอร์ C ซึ่งเราใช้ในการนับเวลา เรากำหนดค่าในรีจีสเตอร์ B ด้วยตัวเลขที่ทำให้เวลาที่ใช้ในการวนรอบ กินเวลา 1 วินาที เสร็จเรียบร้อยแล้วก็จะเริ่มการนับเราทำให้บิท 7 ของเอาต์พุตพอร์ตเป็นหนึ่งแล้วเข้าสู่ส่วนของโปรแกรมที่ทำงานวนรอบ (loop) ทุกครั้งที่เราผ่านเข้าไปในส่วนโปรแกรมวนรอบจำนวนนับในรีจีสเตอร์ B จะลดลงจนกระทั่งเป็น 0 ซึ่งหมายความว่าเวลา 1 วินาทีผ่านพ้นไปแล้วเราจึงกำหนดค่าในรีจีสเตอร์ B ใหม่ เพื่อที่จะใช้ในการทำงานในส่วนของโปรแกรมวนรอบอีกครั้ง ดังนั้นเราจะได้เวลาวินาทีที่กำหนดการนับในรีจีสเตอร์ C โดยค่าในรีจีสเตอร์ C จะลดลงเราจะวนทำไปเรื่อยจนกระทั่งค่าในรีจีสเตอร์ C มีค่าเป็น 0 จึงส่งเอาต์พุต 0 ออกทางบิท 7 ของเอาต์พุตพอร์ต และนำค่าที่เราเก็บในแสตค (Stack) กลับคืนสู่รีจีสเตอร์ตามเดิมแล้วกลับคืนสู่โปรแกรมที่เรียก

เพื่อที่จะใช้ ASM เราต้องสร้างโปรแกรมดังเช่นรูปที่ 7.2 โดยใช้ ED หรือโปรแกรมเอดิเตอร์ (Editor) ตัวอื่นเพื่อสร้างซอร์สโปรแกรม (source program) และไฟล์ที่สร้างต้องเป็นชนิด ASM จากนั้นจึงใช้ ASM แปลโปรแกรมภาษาแอสเซมบลี้เป็นรหัสภาษาเครื่องโดยมีรูปแบบการใช้ดังนี้

ASM D: NAME . PRM โดย PRM เป็นพารามิเตอร์ควบคุมเพื่อผลการแสดงเอาต์พุตซึ่งรับรู้โดย ASM ค่าพารามิเตอร์นี้เราจะศึกษารายละเอียดตอนท้ายบทนี้ D: คือ การบอกให้.ASM ทราบว่า โปรแกรมภาษาแอสเซมบลี้ (ที่มีชนิดของไฟล์เป็น ASM) จะอยู่ที่ไดร์ฟไหน ASM จะแปลโปรแกรมภาษาแอสเซมบลี้และสร้างเอาต์พุตไฟล์ขึ้นสองไฟล์ คือ FILENAME. HEX และ FILENAME.PRN ถ้าเราไม่เลือกพารามิเตอร์ควบคุมก็จะได้ผลอย่างเดียวกันคือมี .HEX ไฟล์และ .PRN ไฟล์เกิดขึ้นเช่นกัน .HEX ไฟล์ประกอบด้วยรหัสภาษาเครื่องที่แปลมาจากโปรแกรมภาษาแอสเซมบลี้และจะอยู่ในรูปเลขฐาน 16 และ .PRN

ไฟล์ประกอบด้วยตัวโปรแกรมภาษาแอสเซมบลี้และรหัสภาษาเครื่องที่เกิดจากการแปลโปรแกรมภาษาแอสเซมบลี้รวมทั้งบอกข้อผิดพลาดของโปรแกรม (เช่นเขียนคำสั่งภาษาแอสเซมบลี้ผิด) พร้อมทั้งมีเลขกำกับบรรทัดด้วย .PRN ไฟล์นั้นเครื่องคอมพิวเตอร์ไม่สามารถเข้าใจได้และเอาไปใช้ในการเอ็กซีคิวไม่ได้ ในระหว่างโปรแกรม ASM แปลโปรแกรมภาษาแอสเซมบลี้เป็นโปรแกรมรหัสภาษาเครื่องนั้น ถ้ามีข้อผิดพลาดเกิดขึ้นจากการเขียนโปรแกรมภาษาแอสเซมบลี้ผิด ASM โปรแกรมจะพิมพ์ข้อผิดพลาดนี้ลงใน .PRN ไฟล์ในขณะเดียวกันก็ส่งออกทางจอด้วย

ดังที่ได้กล่าวมาแล้วข้างต้นว่า ASM โปรแกรมอนุญาตให้ผู้ใช้เว้นไม่ต้องใช้พารามิเตอร์ในการควบคุมการทำงาน ดังนั้นอินพุตและเอาต์พุตไฟล์จะอยู่ในดิสค์ที่กำลังติดต่ออยู่ ถ้ากำหนดพารามิเตอร์สามตัวต่อท้าย FILENAME การกระทำจะทำตามเงื่อนไขของพารามิเตอร์ (ซึ่งปกติจะเป็นที่ระบุชนิดของไฟล์) และเราจะต้องจำไว้ว่าชนิดของอินพุตไฟล์ต้องเป็น .ASM การใช้พารามิเตอร์แทนตำแหน่งชนิดของไฟล์ไม่ได้เปลี่ยนชื่อหรือชนิดของไฟล์ที่ .ASM โปรแกรมมองหา ลักษณะการใช้พารามิเตอร์

FILENAME.XYZ

X, Y, Z เป็นอักษรตัวเดียว ASM โปรแกรมจะเข้าใจตามความหมายการใช้ดังนี้

X: A, B, ...เป็นการบอก ASM โปรแกรมว่า โปรแกรมภาษาแอสเซมบลี้ที่ต้องการแปลเป็นโปรแกรมภาษาเครื่องนั้นอยู่ที่ดิสค์ไดร์ฟไหน

Y: A,B,......เป็นการกำหนดดิสค์ไดร์ฟที่ต้องการให้เขียน .HEX ไฟล์ลงไปหรือเป็นการกำหนดว่าไม่มีการสร้าง .HEX ไฟล์

Z: A,B,C,....เป็นการกำหนดดิสค์ไดร์ฟที่ต้องการให้เขียน .PRN ไฟล์ลงไป

X เป็นการกำหนดแสดง .PRN ไฟล์ออกทางจอแทนที่จะเก็บในดิสค์

Z เป็นการกำหนดว่าไม่มีการสร้าง .PRN ไฟล์

ตัวอย่างการใช้พารามิเตอร์ควบคุมการทำงานของ ASM โปรแกรม เช่น

ASM PROGRAM.ABX

มีความหมายว่าไฟล์ชื่อ PROGRAM (ซึ่งมีชนิดของไฟล์เป็น .ASM) อยู่ในดิสค์ไดร์ฟ A และ .HEX ไฟล์ที่สร้างขึ้นให้เขียนลงในดิสค์ไดร์ฟ B และ .PRN ไฟล์ไม่มีการสร้างลงบนดิสค์แต่ให้แสดงออกทางจอแทน

ถ้าพารามิเตอร์ที่ใช้เป็น .AZZ หมายความว่าต้องการตรวจสอบคำสั่งในโปรแกรมซึ่งจะมีการทำงานเร็วกว่าเนื่องจากไม่ต้องมีการสร้าง HEX และ PRN ไฟล์และข้อผิดพลาดของโปรแกรมจะแสดงออกทางจอ

รูปแบบของโปรแกรม

โปรแกรมภาษาแอสเซมบลี้ซึ่งใช้กับ ASM โปรแกรมจะมีรูปแบบดังนี้

เลขบรรทัด	เลเบล	คำสั่ง	โอเปอแรนด์	; ส่วนขยายความ
0031	Loop	DCR	B	; DCR COUNT

บางส่วนอาจไม่มี (เช่น ในบางบรรทัดอาจไม่มีเลเบล หรือบางคำสั่งที่ใช้อาจไม่ต้องการโอเปอแรนด์ ส่วนขยายความนั้นมีหรือไม่มีก็ได้และแต่ละส่วนต้องมีช่องว่าง (blank) อยู่อย่างน้อย 1 ช่องคั่นอยู่หรืออาจเป็นรหัส ASCII ของ TAB (CNTR-I) ยกเว้นตัวโอเปอร์แรนด์ ซึ่งอาจไม่มีช่องว่างต่อท้ายแต่ต้องปิดท้ายด้วย <cr>หรือรีเทิร์น (ถ้ามีส่วนขยายความตามหลัง) ทุกบรรทัดต้องสิ้นสุดด้วย<cr>หรือ " ! " เพื่อบอก ASM ให้ทราบว่าสิ้นสุดบรรทัดแล้ว

การใช้ ! แทน<cr>เพื่อที่จะให้มีหลายประโยคของภาษาแอสเซมบลีอยู่ในบรรทัดเดียวกันทำให้โปรแกรมง่ายต่อการอ่านและพิมพ์ได้สั้น ๆ ในบางกรณีเช่นโปรแกรมรูปที่ 7.2 เราใช้ "!" กับคำสั่ง POP และ PUSH เพื่อที่จะเก็บรักษาค่าในรีจีสเตอร์ทุกตัวเมื่อเข้าสู่โปรแกรมย่อยและส่งคืนค่าเหล่านี้กลับคืนรีจีสเตอร์ต่าง ๆ ตามเดิมเมื่อจะออกจากโปรแกรมย่อย

PUSH H! PUSH D! PUSH B! PUSH PSW ; เก็บรักษาค่าในรีจีสเตอร์

POP PSW! POP S! POP D! POP H ; นำค่าที่เก็บรักษาไว้ส่งคืนรีจีสเตอร์

ตัวเลขบอกบรรทัดนั้น ASM จะไม่นำมาใช้แต่อนุญาตให้มีเพื่อให้สามารถสร้างโปรแกรมภาษาแอสเซมบลีได้โดยใช้ ED ซึ่งต้องใช้เลขกำกับบรรทัด ดังนั้นโปรแกรมภาษาแอสเซมบลีอาจใช้เอดิเตอร์หรือเวิร์ดโปรเซสซิ่ง (โปรแกรมควบคุมการพิมพ์ตัวอื่นเขียนขึ้นได้โดยไม่ต้องมีเลขกำกับบรรทัด)

เลเบลคือ กลุ่มตัวอักษรและตัวเลขความยาวไม่เกิน 16 ตัวและต้องเริ่มต้นด้วยตัวอักษรและอาจตามหลังด้วยเครื่องหมาย ":" หรือไม่ก็ได้ขึ้นอยู่กับผู้เขียนโปรแกรมและเพื่อที่จะให้เลเบลอ่านง่ายขึ้นเราอาจใช้สัญลักษณ์ "$" แทรกอยู่ในเลเบล

ตัวอย่างของเลเบล

JUMPTORESTART

เพื่อให้อ่านง่าย JUMPTORESTART

สัญลักษณ์ "$" ไม่มีความหมายต่อ ASM โดย "$" ไม่ถูกนับเป็นตัวอักษรตัวหนึ่งของเลเบล ดังนั้นเมื่อนับความยาวของเลเบลจะไม่นับ "$" รวมเข้าไปด้วย

ถัดมาคือ ส่วนของคำสั่งซึ่งจะเป็นคำสั่งภาษาแอสเซมบลีหรือคำสั่งเทียมใช้ในการบอกการทำงานที่ต้องการและบางคำสั่งจะต้องการโอเปอร์แรนด์ โอเปอร์แรนด์จะอยู่ฟิลด์ถัดมาอาจเป็นค่าคงที่ เลเบลที่อ้างถึงโดยคำสั่ง เงื่อนไขตรรกศาสตร์หรือการบวกลบทางคณิตศาสตร์ ซึ่ง ASM จะคำนวณค่าให้

ตัวอย่าง

```
EQU   0F0FH
OUT   OUTPORT
MVI   B,L + TIMER
```

ฟิลด์สุดท้ายเรียกว่า *ส่วนขยายความ (COMMENT)* ฟิลด์หรือส่วนขยายความใช้อธิบายการทำงานของแต่ละบรรทัดที่ต้องการและ ASM จะไม่นำส่วนขยายความมาทำการแปล ประโยชน์ของส่วนขยายความนี้ทำให้ง่ายต่อการทำความเข้าใจโปรแกรมเมื่อมาอ่านภายหลังหรือง่ายต่อบุคคลอื่นที่มาอ่านโปรแกรม

เนื่องจากโปรแกรมภาษาแอสเซมบลีมีโครงสร้างน้อยกว่าภาษาระดับสูง ดังนั้นจึงมีลักษณะการทำงานได้กว้างกว่า ASM จะทำการแปลโปรแกรมภาษาแอสเซมบลีจนกระทั่งพบรหัสบอกการสิ้นสุดของ

ไฟล์หรือคำสั่ง END ASM จะไม่ทำการแปลข้อความที่อยู่ข้างท้ายคำสั่ง END นี้

ค่าคงที่ตัวเลข คือ ตัวเลข 8 บิทหรือ 16 บิทที่ใช้ในโปรแกรมหรืออาจแสดงได้โดยใช้เลขฐานต่าง ๆ (เช่นฐาน 2 ฐาน 8 ฐาน 16) เพื่อให้เหมาะกับการใช้งาน เช่น ตำแหน่งในหน่วยความจำจะใช้เลขฐาน 16 และการนับใช้เลขฐาน 10 เป็นต้น และเพื่อให้ ASM รู้ฐานของตัวเลขที่ใช้เราจะใช้อักษรแสดงข้างล่างนี้เป็นตัวบอก

B = เลขฐาน 2
O = เลขฐาน 8
Q = เลขฐาน 8
D = เลขฐาน 10
H = เลขฐาน 16

เหตุผลที่เลขฐาน 8 ใช้ตัวอักษรบอกถึง 2 ตัว เนื่องจากว่าอักษร O อาจง่ายต่อการสับสนกับเลข 0 ASM จะตีความตัวเลขที่ไม่มีอักขระระบุฐานเลขว่าเป็นเลขฐาน 10 และในกรณีเลขฐานสิบหกค่าคงที่ต้องเริ่มต้นด้วยตัวเลข ถ้าเลขตัวแรกเป็นตัวอักษรต้องมีเลขศูนย์นำหน้า ASM จะได้ไม่ตีความว่าเป็นตัวหนังสือหรือค่าตัวแปรอื่น

เลขฐาน 2	11110000B
	1010$1000B
เลขฐาน 8	2267O
	3325Q
	22$56$12Q
เลขฐาน 10	128
	48354D
เลขฐาน 16	0FFH
	81H
	0DEFH

ค่าคงที่สตริง คือสัญลักษณ์ ASCII ที่เรียงกันไม่เกิน 64 ตัว และไม่มีรหัส ASCII ที่ใช้เป็นสัญลักษณ์ในการควบคุมการทำงาน เช่น cr, CTRL - Z, CTRL - C เป็นต้น (รหัส 32 ตัวแรกในตาราง ASCII) เราทราบได้ว่าข้อความใดเป็นค่าคงที่สตริงโดยมีเครื่องหมาย ' ปิดหัวและท้ายข้อความ

ASM จะไม่เปลี่ยนอักษรตัวเล็กในค่าคงที่สตริง เป็นอักษรตัวใหญ่แต่ถ้าต้องการให้มีเครื่องหมาย ' ปรากฏของในข้อความด้วยต้องใช้เครื่องหมาย '' เมื่อ ASM พบก็จะได้รู้ว่ายังไม่สิ้นสุดข้อความของสตริง

ตัวอย่าง

'ERROR CODE 96'

'Please insert the diskette in drive A:'

'I said ''Good Morning'' to him'

คำสั่งคณิตศาสตร์และตรรกศาสตร์ (ลอจิก)

ดังที่กล่าวมาแล้วข้างต้นว่าโอเปอร์แรนด์อาจเป็นค่าคงที่หรือลอจิกหรือตัวแปรทางคณิตศาสตร์ ทั้งนี้เพื่อให้มีประสิทธิภาพและง่ายต่อการเขียนโปรแกรมซึ่งทำให้โปรแกรมมีลักษณะง่ายต่อการอ่าน ตัวกระทำทางคณิตศาสตร์และลอจิกอาจใช้ร่วมกันเพื่อประกอบเป็นข้อความทางคณิตศาสตร์ได้

ตารางที่ 7-1 จะแสดงให้เห็นถึงตัวกระทำทางคณิตศาสตร์และลอจิกที่ ASM เข้าใจได้

a + b	หาผลบวกของ a และ b โดยไม่แสดงเครื่องหมาย
a − b	หาผลต่างของ a และ b โดยไม่แสดงเครื่องหมาย
+ b	ค่า b กำหนดเป็นบวก
− b	ค่า b กำหนดเป็นลบ
a * b	หาผลคูณของ a และ b โดยไม่แสดงเครื่องหมาย
a/b	หาผลหารของ a และ b โดยไม่แสดงเครื่องหมาย
a MOD b	หาเศษเหลือจากการหารของ a/b
NOT b	กลับเงื่อนไขทางตรรกศาสตร์ของ b (0 ทุกตัวเปลี่ยนเป็น 1,1 ทุกตัวเปลี่ยนเป็น 0)
a AND b	การทำงานทางตรรกศาสตร์ AND บิทต่อบิทของ a และ b
a OR b	การทำงานทางตรรกศาสตร์ OR บิทต่อบิทของ a และ b
a XOR b	การทำงานทางตรรกศาสตร์ EXCLUSIVE OR ของ a และ b
a SHL b	การเลื่อนค่าในบิทต่าง ๆ ของ a ไปทางซ้ายกำหนดโดยค่าใน B บิทที่ไม่มีค่าแทนที่จะถูกแทนที่ด้วย
a SHR b	การเลื่อนตำแหน่งของบิทต่าง ๆ ของ a ไปทางขวากำหนดโดยค่าใน B บิทที่ไม่มีอะไรแทนที่จะถูกแทนที่ด้วย 0

ในโปรแกรมภาษาแอสเซมบลีที่เขียนขึ้นอาจมีส่วนของโปรแกรมที่ใช้เป็นที่เก็บข้อมูลซึ่งส่วนของโปรแกรมนี้ ASM ไม่ต้องแปลส่วนของโปรแกรมนี้โดยใช้คำสั่งกระโดดข้ามไป

JMP ENTRY + 4 ; คำสั่งกระโดดข้ามไปจาก ENTRY + 4 ถ้า ENTRY คือ 0D800H แล้ว ASM จะตีความเป็น 0 D800H + 04 หรือ 0D804H โปรดจำไว้เสมอว่า ASM จะทำงานกับตัวเลข 16 บิท ดังนั้นถ้าโอเปอร์แรนด์เป็น DDH + OFH จะมีค่าเป็น 00ECH ไม่ใช้ ECH

ผลบวกของโอเปอร์แรนด์นี้จะเข้าไปอยู่ในคู่รีจีสเตอร์ของ 8080 ซึ่งมี 16 บิทไม่ใช่เข้าไปอยู่ในรีจีสเตอร์ 8 บิท ASM จะตีตามความสัมพันธ์ของโอเปอร์แรนด์ที่แสดงข้างบนซึ่งแยกออกได้ 5 ระดับโดยการคูณจะมีระดับสูงสุดโดยคิดจากทางซ้ายไปขวาและถ้ามี ( ) แทรกอยู่ ASM จะตีความภายในวงเล็บก่อนตามตารางข้างล่างนี้การกระทำในบรรทัดเดียวกันจะมีความสำคัญเท่ากันและมีความสำคัญมากกว่าบรรทัดที่อยู่ถัดไป

* / MOD SHL SHR

− +

NOT

AND

OR XOR

ตัวอย่าง

a ∗ b + c ∗ d จะมีความหมายเหมือนกับ (a ∗ b) + (c ∗ d) กล่าวคือ ASM จะหาผลคูณของ a∗b ก่อนตามด้วย c∗d จากนั้นจึงจะทำการบวกกัน

คำสั่งเทียมของ ASM

จะมีคำสั่งบางคำสั่งซึ่งไม่อยู่ในชุดคำสั่งของ 8080 แต่ ASM สามารถเข้าใจได้ คำสั่งเหล่านี้เรียกว่าคำสั่งเทียมใช้ในการจัดการเกี่ยวกับส่วนของข้อมูลในหน่วยความจำซึ่งไม่ใช่ที่เก็บคำสั่ง เช่นใช้ในการจองที่ของตัวแปรหรือกำหนดการเริ่มต้นหรือสิ้นสุดของตำแหน่งในหน่วยความจำของรหัสที่สร้างโดย ASM ตัวอย่างของคำสั่งเทียมได้แก่

ORG	บอกตำแหน่งเริ่มต้นของโปรแกรมหรือข้อมูล
END	บอกตำแหน่งสิ้นสุดของโปรแกรมแอสเซมบลี้
EQU	ใช้กำหนดค่าคงที่ตัวเลขให้กับตัวแปร
SET	เซตค่าตัวเลข
IF	เริ่มต้นการตัดสินใจ
ENDIF	สิ้นสุดการตัดสินใจ
DB	กำหนดข้อมูลมีความยาวเป็นไบท์ (8 บิท)
DW	กำหนดข้อมูลมีความยาวเป็นเวิร์ด (16 บิท)
DS	กำหนดตำแหน่งเริ่มต้นเก็บข้อมูลและจำนวนข้อมูลที่จะเก็บ

การใช้คำสั่งเทียมก็เช่นเดียวกับคำสั่งของ 8080 ไมโครโปรเซสเซอร์โดยมีลักษณะการใช้ดังต่อไปนี้

ORG : ย่อมาจากคำว่า origin ซึ่งแปลว่าจุดเริ่มต้นเป็นการบอกตำแหน่งเริ่มต้นว่าอยู่ตำแหน่งไหนหรือบอกจุดเริ่มต้นของข้อมูลมีรูปแบบการใช้ดังนี้

เลเบล ORG expression ; ส่วนขยายความ

เลเบลหรือส่วนขยายความอาจมีหรือไม่มีก็ได้ เอ็กซ์เปรสชั่น (expression) คือค่าตัวเลข 16 บิท หรือมีค่าเท่ากับเลข 16 บิท (เพราะการกำหนดตำแหน่งของหน่วยความจำต้องการ 16 บิท 8 บิท ไม่พอ)

ORG จะนำหน้าก่อนประโยคภาษาแอสเซมบลี้ใด ๆ ที่ต้องการแปลเป็นประโยคภาษาเครื่องซึ่งเมื่อแปลแล้วจะเริ่มต้นที่ตำแหน่งในหน่วยความจำที่ระบุโดย ORG

โปรแกรมแต่ละโปรแกรมอาจมี ORG ได้หลายตัวเพื่อกำหนดส่วนของโปรแกรมที่ต้องการให้อยู่ในหน่วยความจำตำแหน่งที่ต้องการ ตามปกติโปรแกรมภาษาเครื่องที่ทำงานภายใต้ CP/M จะเริ่มต้นที่หน่วยความจำ 100H ดังนั้นโปรแกรมควรมีประโยค

ORG 0100H

แต่ถ้าเราเว้นคำสั่งนี้ ASM จะกำหนดให้โปรแกรมเริ่มต้นที่ 0000H และไม่ทำงานภายใต้ CP/M เพราะว่าหน่วยความจำตั้งแต่ 0000H ถึง 0100H นั้นเป็นหน่วยความจำที่ใช้โดย CP/M

ตัวอย่างของโปรแกรมตามรูปที่ 7.2 นั้นเริ่มต้นที่ D800H เพราะว่ามันต้องทำงานร่วมกับโปรแกรมภาษาเบสิคและตัวโปรแกรมภาษาเบสิคจะเริ่มต้นทำงานที่ 0100H เราจึงหาตำแหน่งอื่นเพื่อเก็บโปร

แกรมภาษาเครื่องโดยใช้คำสั่ง ORG

ORG 0D800H

END : END เป็นคำสั่งแสดงจุดสิ้นสุดของโปรแกรมโดย ASM จะหยุดการแแปลที่จุดนี้และไม่สนใจว่ามีบรรทัดต่อมาหรือไม่มี รูปแบบการใช้ดังนี้

เลเบล END

โดยเลเบลอาจมีหรือไม่มีก็ได้แล้วแต่การเลือกใช้

EQU : EQU ย่อมาจากคำว่า EQUAL ซึ่งแปลว่าเท่ากันมีลักษณะการใช้สองอย่างคือ อย่างแรกใช้กำหนดค่าของเลเบลดังนั้นเลเบลจึงสามารถใช้ได้ตลอดทั้งโปรแกรมและการทำเช่นนี้ทำให้โปรแกรมง่ายต่อการอ่านยิ่งขึ้น เนื่องจากเลเบลจะให้ความดังใจได้ความว่าค่าคงที่ตัวเลขที่ใช้ในงานอย่างเดียวกันมีรูปแบบการใช้ดังนี้

เลเบล EQU expression

โดยเลเบลคือกลุ่มตัวอักษรที่เรากำหนดและเอ็กซ์เปรสชัน (expression) จะใช้กำหนดค่าของเลเบล ตัวอย่างการใช้ เช่นเราต้องการสร้างหน่วยความจำสำรอง (buffer) เริ่มต้นที่ตำแหน่ง 0D00H

BUFFER EQU 0D00H

เมื่อเราต้องการอ้างอิงถึงภายหลังในโปรแกรมก็นำมาเก็บใน HL รีจีสเตอร์โดยคำสั่งข้างล่างนี้

LXI H, BUFFER

แทนที่จะเป็น

LXI H, 0D00H

ซึ่งก็มีผลเป็นรหัสภาษาเครื่องเหมือนกันและถ้าเราต้องการเปลี่ยนตำแหน่งเริ่มต้นของ BUFFER ก็เพียงแต่เปลี่ยนแต่บรรทัดเดียวเท่านั้น ถ้าเราใช้ EQU เลเบลจะไม่สามารถกำหนดเป็นค่าอื่นได้ภายหลังถ้าต้องการเปลี่ยนสามารถทำได้โดยคำสั่ง SET

SET : SET มีลักษณะการใช้งานเหมือน EQU ต่างกันตรงที่ว่าคำสั่ง SET สามารถเปลี่ยนค่าของเลเบลที่กำหนดค่าโดยคำสั่ง EQU ได้ คำสั่ง SET ส่วนมากใช้ในการควบคุมเงื่อนไขการทำงานซึ่งจะอธิบายในภายหลัง

IF และ **ENDIF** IF และ ENDIF อนุญาตให้เราสามารถแยกบางส่วนของโปรแกรมและเลือกทำงานหรือไม่ทำงานตามต้องการแล้วแต่เงื่อนไขที่กำหนดในเอ็กซ์เปรสชันจะเป็นจริงหรือไม่จริงถ้าเป็นจริงก็จะทำ

```
เลเบล IF   expression
           คำสั่งที่ # 1
           คำสั่งที่ # 2
           คำสั่งที่ # w
ENDIF
```

ตามตัวอย่าง ASM จัดว่าเงื่อนไขเป็นจริงทางตรรกศาสตร์ (0 หรือ -1) ของเอกซ์เปรสชั่น (expression) ถ้าเป็นจริงมันจะทำตามคำสั่ง คำสั่งที่#1 ถึง#n ถ้าไม่จริง ASM จะข้ามคำสั่งที่ 1 ถึง n และเริ่ม

ทำงาน คำสั่งแรกต่อจาก ENDIF

IF และ ENDIF ปกติใช้ในการทดสอบและหาข้อผิดพลาดของโปรแกรม ยกตัวอย่าง เช่น เราต้องการหาข้อผิดพลาดของโปรแกรมที่ทำงานมีลักษณะวนรอบ (loop) และต้องการรู้ว่าก่อนที่จะออกจากส่วนของโปรแกรมที่ทำงานรอบนี้โปรแกรมทำงานวนรอบกี่ครั้ง

```
True     EQU  -1
Fals     EQU  0
Test     EQU  Trur ;  กำหนด test เป็นเงื่อนไขจริง
         :
         MVI  A, 10  ;  จำนวนครั้งการวนรอบ
         MVI  B, 0   ;  เคลียร์รีจีสเตอร์ B
Loop :   DCR  B      ;  ลดค่ารีจีสเตอร์ B ลง 1
         IF   test   ;  เริ่มต้นส่วนของโปรแกรมทดสอบ
         INC  B  เพิ่มค่าตัวนับของส่วนของโปรแกรมทดสอบ
         ENDIF
         JNZ  Loop ;  กลับไม่เริ่มต้นใหม่ถ้ารีจีสเตอร์ $A ยังไม่เป็นศูนย์
```

เมื่อจบการทำงานค่าในรีจีสเตอร์ B จะบอกจำนวนครั้งของการวนรอบถ้าเราต้องการยกเลิกการทำงานทดสอบก็โดยเปลี่ยนอย่างง่าย ๆ โดยเปลี่ยนจาก test EQU True เป็น

Test EQU False ; ยกเลิกการทดสอบ

อย่างไรก็ตามเราสามารถเรียกใช้ได้อีกโดยการเปลี่ยน test เป็น true เมื่อต้องการทดสอบอีกได้ในภายหลัง

DB, DW, DS DB, DW, DS เป็นการกำหนดตำแหน่งเริ่มต้นของหน่วยความจำโดย DB กำหนด 8 บิทต่อ 1 ค่า แต่ต่อเนื่องไปเรื่อย ๆ DW ก็มีการทำงานเหมือน DB เพียงแต่กำหนด 16 บิทต่อ 1 ค่าอย่างต่อเนื่อง มีรูปแบบการใช้ดังนี้

```
เลเบล DB V#1, V#2,......,V#n
เลเบล DW  V#1, V#2,......,V#n
เลเบล DS  expression
```

เลเบลอาจมีหรือไม่มีก็ได้ V#1 ถึง V#n เป็นค่า 8 บิทกำหนดโดย DB และเป็นค่า 16 บิทกำหนดโดย DW และค่าใน DB อาจเป็นสัญลักษณ์สตริงได้ดังตัวอย่าง

```
Emessage DB 'you blew it'
```

ASM โปรแกรมจะเก็บอักษรแต่ละตัวด้วยรหัส ASCII เรียงไปในหน่วยความจำตามลำดับและพึงระลึกได้เสมอว่า 8080, 8085 ไมโครโปรเซสเซอร์จะเก็บค่า 16 บิทในหน่วยความจำโดยไบท์ ส่วนที่มีนัยสำคัญน้อยกว่าเก็บในตำแหน่งความจำต่ำกว่าของคู่หน่วยความจำที่อ้างถึงและไบท์ที่มีนัยสำคัญมากกว่าจะเก็บในหน่วยความจำตำแหน่งถัดไปแต่เราไม่จำเป็นต้องกังวลในเรื่องนี้เนื่องจาก ASM จะจัดการให้โดยอัตโนมัติ

DS จะจัดค่าเอ็กซ์เปรสชั่นและเก็บรักษาค่าตัวเลขที่ระบุ (ไบท์) ในหน่วยความจำและใช้เป็นที่เก็บ

ข้อมูลสำรองเรียกใช้โดยโปรแกรม โดยไม่ต้องกำหนดค่าตั้งแต่ต้น ตัวอย่างข้างล่างนี้จะสำรองที่ไว้ 128 ไบท์ของหน่วยความจำสำหรับเป็นหน่วยความจำสำรอง (Buffer) เพื่อใช้กับ I/O ดิสค์

Diskbuff DS 128 ; สำรองที่ 128 ไบท์

ตัวอย่างการใช้ ASM

การใช้คำสั่ง ASM FILEASM.AAZ จะนำโปรแกรมหลัก (Source Program) ชื่อ FILEASM. AAZ ซึ่งอยู่ในดิสค์ไดร์ฟ A แปลงเป็นโปรแกรมภาษาเครื่องโดยกำหนด .HEX ไฟล์เก็บไว้ในดิสค์ไดร์ฟ A และ .PRN ไฟล์จะแสดงออกทางจอแทนที่จะเก็บในดิสค์ (PRN ไฟล์จะเก็บทั้งตัวโปรแกรมภาษาแอสเซมบลี้และโค้ดภาษาเครื่องรวมอยู่ด้วยกันรวมทั้งตำแหน่งของโปรแกรมในหน่วยความจำด้วย) และการแก้ไขปรับปรุงและทดสอบเราจะใช้โปรแกรม DDT ของ CP/M ซึ่งจะอธิบายในบทต่อไป

```
A>
A>ASM FILEASM.AAX
CP/M ASSEMBLER - VER 2.0


                ;PROGRAM FOR DEMONSTRATION
                ;DATE:      MARCH 1,1984
                ;
8000                  ORG     8000H     ;ORIGIN PROGRAM AT 8000H
00A1 =          OUTPUT: EQU   0A1H      ;OUTPUT PORT NUMBER
0001 =          ON:     EQU   01H       ;TURN ON BIT 0
0064 =          LTIME:  EQU   100       ;1MS LOOP COUNT
                ;
8000 C30480     START:  JMP   MAIN
8003 64         COUNT:  DB    100       ;SET COUNT AT THIS LOCATION(100MS)
                ;
8004 E5         MAIN:   PUSH  H         ;SAVE ALL REGISTERS
8005 D5                 PUSH  D
8006 C5                 PUSH  B
8007 F5                 PUSH  PSW
8008 3E00               MVI   A,0       ;CLEAR ACCUMULATOR
800A D3A1               OUT   OUTPUT    ;SET BIT 0 TO 0
800C 3A0380             LDA   COUNT     ;GET X MS COUNT
800F 4F                 MOV   C,A       ;SAVE IT
8010 0664               MVI   B,LTIME   ;LOOP COUNT = 1 MS
8012 3E01               MVI   A,ON      ;SET BIT 0 TO 1
8014 D3A1               OUT   OUTPUT    ;START PERIOD
                        ;
8016 05         LOOPT:  DCR   B         ;DECREMENT LOOP COUNT
8017 00                 NOP             ;DELAY
8018 00                 NOP
8019 00                 NOP
801A 00                 NOP
801B C21680             JNZ   LOOPT     ;LOOP IF NOT OK
                ;
801E 0664               MVI   B,LTIME   ;RESET LOOP COUNT
8020 0D                 DCR   C         ;DECREMENT 1 MS
8021 C21680             JNZ   LOOPT     ;GO AGAIN
                ;
8024 3E00               MVI   A,0       ;CLEAR ACCUMULATOR
8026 D3A1               OUT   OUTPUT    ;END PERIOD
                ;
8028 F1                 POP   PSW       ;RESTORE REGISTER
```

```
 8029 C1            POP    B
 802A D1            POP    D
 802B E1            POP    H
                ;
 802C C9            RET            ;RETURN TO MAIN PROGRAM
 802D               END
802D
000H USE FACTOR
END OF ASSEMBLY
```

เมื่อเสร็จสิ้นการแอสเซมเบลอร์แบบนี้แล้วจะเห็นภาพปรากฏบนจอในลักษณะข้างบนที่กล่าวมาแล้ว แต่จะไม่มี FILE ASM. PRN เกิดขึ้น สิ่งที่เกิดขึ้นจะเกิดขึ้นเฉพาะไฟล์ FILE ASM. HEX เท่านั้น เมื่อ DIR ดูจะได้ดังนี้

```
A>
A>
A>DIR
A: FORMAT   COM : COPY     BAK : MBASIC   COM : CPM      COM
A: SIRI     ASM : CONFIGIO BAS : PIP      COM : STAT     COM
A: ED       COM : ASM      COM : DDT      COM : LOAD     COM
A: RW13     COM : APDOS    COM : SUBMIT   COM : XSUB     COM
A: DUMP     BAK : DUMP     COM : DOWNLOAD COM : CPM56    COM
A: UPLOAD   COM : SIRI     $$$ : SIRI     PRN : SIRI     HEX
A: FORMAT   ASM : FORMAT   $$$ : COPY     COM : COPY     ASM
A: FILEASM  BAK : PRN      HEX : DUMP     ASM : LOV
A: LOV      $$$ : FILEASM  ASM : FILEASM  HEX : FILE     PRN
A: FILE     HEX
```

แต่หากเรียกการแอสเซมเบลอร์โดยใช้ ASM FILEASM โดยไม่มีพารามิเตอร์ตามมาผลที่ได้คือไมโครคอมพิวเตอร์จะไม่แสดงผลออกมาบนจอ แต่ถ้ามีข้อผิดพลาดจะแสดงออกมา

```
A>
A>ASM FILEASM
CP/M ASSEMBLER - VER 2.0
802D
000H USE FACTOR
END OF ASSEMBLY
```

เมื่อได้ทำการแอสเซมเบลอร์เรียบร้อยแล้ว ผลที่เกิดขึ้นคราวนี้จะมีไฟล์ FILEASM.HEX และไฟล์ FILEASM.PRN เกิดขึ้น ซึ่งไฟล์ .PRN นี้สามารถนำมาพิมพ์ออกทางเครื่องพิมพ์

```
A>
A>DIR
A: FORMAT   COM : COPY     BAK : MBASIC   COM : CPM      COM
A: SIRI     ASM : CONFIGIO BAS : PIP      COM : STAT     COM
A: ED       COM : ASM      COM : DDT      COM : LOAD     COM
A: RW13     COM : APDOS    COM : SUBMIT   COM : XSUB     COM
A: DUMP     BAK : DUMP     COM : DOWNLOAD COM : CPM56    COM
A: UPLOAD   COM : SIRI     $$$ : SIRI     PRN : SIRI     HEX
```

```
A: FORMAT   ASM : FORMAT   $$$ : COPY    COM : COPY    ASM
A: FILEASM  BAK : PRN      HEX : DUMP    ASM : LOV
A: LOV      $$$ : FILEASM  ASM : FILEASM HEX : FILE    PRN
A: FILE     HEX : FILEASM  PRN
A>
```

หากต้องการพิมพ์ก็ใช้คำสั่ง TYPE FILEASM.PRN และถ้าต้องการออกเครื่องพิมพ์ก็ตามด้วย CTRL-P ผลที่ได้จะเป็นดังนี้

```
A>TYPE FILEASM.PRN

                  ;PROGRAM FOR DEMONSTRATION
                  ;DATE:   MARCH 1,1984
                  ;
 8000                       ORG     8000H    ;ORIGIN PROGRAM AT 8000H
 00A1 =           OUTPUT:   EQU     0A1H     ;OUTPUT PORT NUMBER
 0001 =           ON:       EQU     01H      ;TURN ON BIT 0
 0064 =           LTIME:    EQU     100      ;1MS LOOP COUNT
                  ;
 8000 C30480      START:    JMP     MAIN
 8003 64          COUNT:    DB      100      ;SET COUNT AT THIS LOCATION(100MS)
                  ;
 8004 E5          MAIN:     PUSH    H        ;SAVE ALL REGISTERS
 8005 D5                    PUSH    D
 8006 C5                    PUSH    B
 8007 F5                    PUSH    PSW
                  ;
 8008 3E00                  MVI     A,0      ;CLEAR ACCUMULATOR
 800A D3A1                  OUT     OUTPUT   ;SET BIT 0 TO 0
 800C 3A0380                LDA     COUNT    ;GET X MS COUNT
 800F 4F                    MOV     C,A      ;SAVE IT
 8010 0664                  MVI     B,LTIME  ;LOOP COUNT = 1 MS
 8012 3E01                  MVI     A,ON     ;SET BIT 0 TO 1
 8014 D3A1                  OUT     OUTPUT   ;START PERIOD
                  ;
 8016 05          LOOPT:    DCR     B        ;DECREMENT LOOP COUNT
 8017 00                    NOP              ;DELAY
 8018 00                    NOP
 8019 00                    NOP
 801A 00                    NOP
 801B C21680                JNZ     LOOPT    ;LOOP IF NOT OK
                  ;
 801E 0664                  MVI     B,LTIME  ;RESET LOOP COUNT
 8020 0D                    DCR     C        ;DECREMENT 1 MS
 8021 C21680                JNZ     LOOPT    ;GO AGAIN
                  ;
 8024 3E00                  MVI     A,0      ;CLEAR ACCUMULATOR
 8026 D3A1                  OUT     OUTPUT   ;END PERIOD
                  ;
 8028 F1                    POP     PSW      ;RESTORE REGISTER
 8029 C1                    POP     B
 802A D1                    POP     D
 802B E1                    POP     H
                  ;
 802C C9                    RET              ;RETURN TO MAIN PROGRAM
 802D                       END
```

8

DDT

โดยปกติโปรแกรมภาษาแอสเซมบลี้ที่เขียนขึ้นมักมีข้อผิดพลาดและยากต่อการค้นหาและแก้ไขส่วนที่ผิด ความยากง่ายของการค้นหาและแก้ไขส่วนที่ผิดนั้นยังขึ้นอยู่กับความซับซ้อนของโปรแกรมอีกด้วย ดังนั้นจึงมีความจำเป็นที่จะต้องมีเครื่องมือช่วยในการแก้ไขและปรับปรุงโปรแกรมตามที่ต้องการ DDT (dynamic debuging tool) เป็นเครื่องมือที่จะช่วยในการตรวจสอบแก้ไขข้อผิดพลาดได้เป็นอย่างดี

DDT ได้รับการเขียนขึ้นเพื่อทำงานกับโปรแกรมภาษาแอสเซมบลี้ของไมโครโปรเซสเซอร์ 8080 8085 เพื่อการตรวจสอบและปรับปรุงข้อมูลในหน่วยความจำตำแหน่งต่าง ๆ และ/หรือค่าในรีจีสเตอร์ของซีพียู และสามารถควบคุมให้โปรแกรมทำงานทีละสเต็ปเพื่อดูผลการทำงานของโปรแกรมโดยยอมให้ผู้ใช้นำโปรแกรมที่มีชนิดของไฟล์เป็น HEX ซึ่งเกิดจากการแอสเซมเบลอร์ด้วย ASM มาใช้ทำงานกับเครื่องคอมพิวเตอร์ได้

ส่วนประกอบของ DDT

DDT มีโปรแกรมที่แยกออกจากกันได้ 2 ส่วนหรืออาจเรียกว่า 2 โมดูลส่วนแรกเป็นหัวใจของ DDT ใช้ในการอินเตอร์พลีต (เป็นการแปลภาษาแอสเซมบลี้เป็นภาษาเครื่องทีละประโยคแล้วนำไปทำงานกับเครื่องคอมพิวเตอร์เสร็จแล้วจึงทำการแปลประโยคภาษาแอสเซมบลี้บรรทัดต่อไปทำไปเรื่อยจนจบโปรแกรม) คำสั่งของ DDT ยกเว้นคำสั่ง A (แอสเซมเบลอร์) และ L (ดิสแอสเซมเบลอร์) เป็นคำสั่งของโปรแกรมส่วนที่สองซึ่งจะกล่าวรายละเอียดในภายหลัง

ส่วนแรกของ DDT จะได้รับการนำมาจากดิสค์เข้ามาในหน่วยความจำใต้ส่วน BDOS ของ CP/M และส่วนที่สองจะอยู่ถัดมาทำให้ CCP ได้รับการแทนที่จนไม่สามารถทำงานด้วย CCP ได้ ดังนั้นเมื่อใช้ DDT แล้วจึงต้องวอร์มบูท เพื่อนำส่วนของ CCP จากดิสค์กลับลงสู่ที่เดิม

การใช้งานโดยการใช้คำสั่งใน DDT

เมื่อเริ่มการใช้งาน ต้องดูว่าในแผ่น CP/M ที่ใช้งานมีโปรแกรม DDT.COM หรือไม่ ถ้ามีการเรียกโปรแกรมจะใช้การเรียกดังนี้

```
A>DDT
```

DDT จะตอบรับโดยข้อความ

```
DDT VERS 2.2
```

โดย DDT ที่เราใช้อยู่จะมีเวอร์ชั่น 2.2

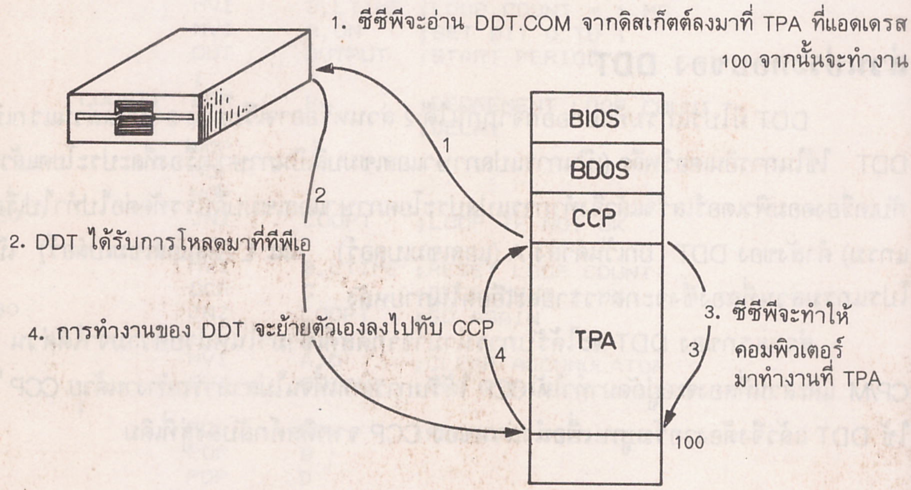
บรรทัดถัดมาจะมีลักษณะ " - "

ซึ่งบอกให้ทราบว่า DDT ทำงานอยู่ในส่วนรับคำสั่งและรอคำสั่งจากผู้ใช้

```
A>
A>DDT
DDT VERS 2.2
-
```

เมื่อมีเครื่องหมาย - แสดงถึงการที่ส่วนของ DDT ได้รับการอ่านจากดิสค์มาเก็บในหน่วยความจำตำแหน่งเดียวกับ CCP

หมายเหตุ CCP จะอ่าน DDT เข้ามาทับตัวมันเองเพราะฉะนั้นจึงต้องอ่าน CCP กลับเข้ามาอีกครั้งจากดิสค์ด้วยการวอร์มบูทเมื่อ DDT ทำงานเสร็จ CCP จะอ่าน DDT. COM จากดิสค์มาเก็บในหน่วยความจำ ผลลัพธ์จะเป็นดังรูปที่ 8.1 จะเห็นได้ว่า DDT ไม่ได้ถูกนำมาเก็บในหน่วยความจำโดยเริ่มต้นที่ 0100H เหมือนไฟล์ชนิด COM อื่น ๆ จึงสามารถใช้ในการปรับปรุงแก้ไขโปรแกรมที่อยู่ใน TPA ได้รวมทั้งตำแหน่งอื่น ๆ ได้ด้วยเช่นในส่วนของ BIOS หรือ BDOS ของ CP/M



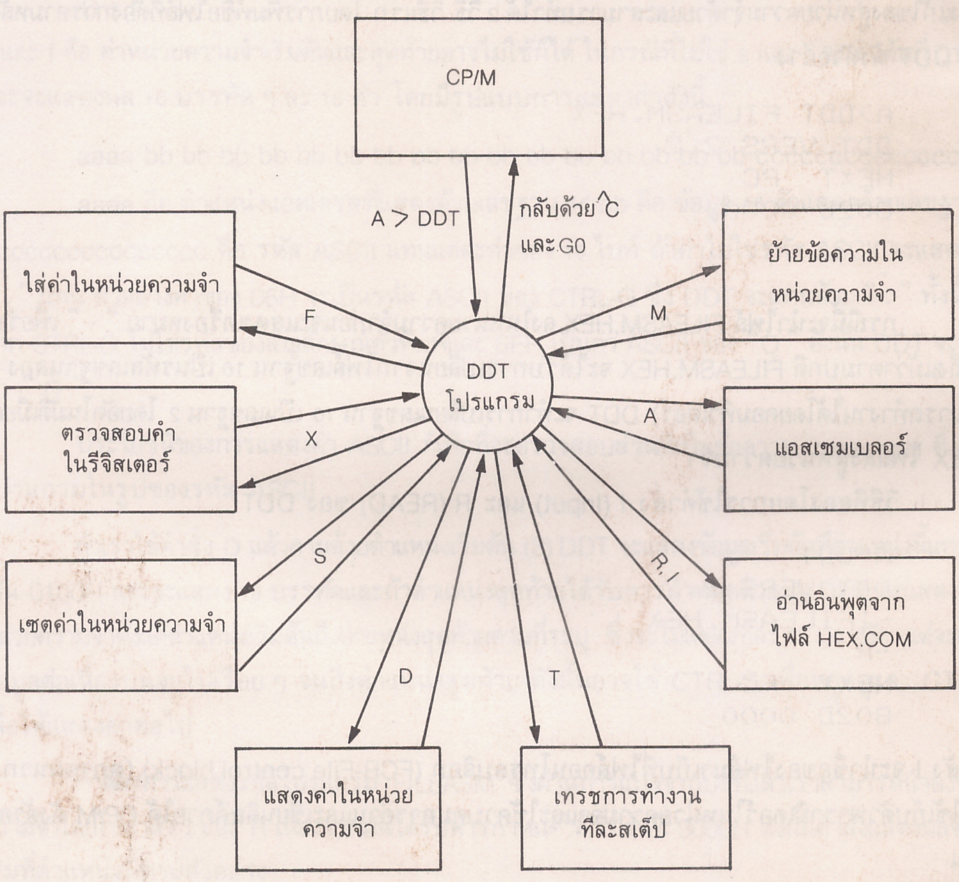
รูปที่ 8.1 ลำดับขั้นการทำงานของ DDT.

ในตอนแรก DDT จะได้รับการนำลงมาในหน่วยความจำตำแหน่งที่ 0100H ก่อนแล้วจึงเคลื่อนย้ายไปอยู่ในตำแหน่งของ CCP ดังนั้นโปรแกรมจะตรวจสอบโดย DDT จะต้องนำมาลงในหน่วยความจำหลัง DDT

เมื่อ DDT ถูกนำลงมาในหน่วยความจำเรียบร้อยแล้วจะเปลี่ยนตำแหน่งของหน่วยความจำสำรองที่ใช้โดย CP/M (ใต้ตำแหน่ง 0100H) โดยตำแหน่งแรกคือหน่วยความจำตำแหน่ง 0006H และ 0007H ซึ่งใช้เก็บตำแหน่งที่ใช้ในการกระโดดไปตำแหน่งในหน่วยความจำตำแหน่งแรกของ BIOS ของ CP/M เพื่อที่จะให้แน่ใจว่า DDT จะไม่ถูกเขียนทับจึงเปลี่ยนตำแหน่งหน่วยความจำในไบท์ 6 และ 7 เป็นตำแหน่งแรกของส่วนแรกของ DDT

อย่างไรก็ตามส่วนของแอสเซมเบลอร์และดิสแอสเซมเบลอร์สามารถได้รับการเขียนทับได้โดยโปรแกรมอื่น ๆ ดังนั้นเพื่อที่จะรักษาส่วนของแอสเซมเบลอร์และดิสแอสเซมเบลอร์ของ DDT ผู้ใช้จึงต้องเปลี่ยนไบท์ 6 และ 7 ด้วยตัวเองขณะใช้ DDT เปลี่ยนค่าเป็นตำแหน่งหน่วยความจำใต้ส่วนโปรแกรมแอสเซมเบลอร์และดิสแอสเซมเบลอร์ของ DDT ด้วยคำสั่งบางคำสั่งของ DDT ซึ่งจะอธิบายต่อไปภายหลัง

ข้อความในหน่วยความจำ 038H ถึง 03AH ซึ่งเป็นตำแหน่งเมื่อทำคำสั่ง RST 7 ซึ่งข้อความนี้คือตำแหน่งที่กระโดดไปเมื่อคำสั่ง RST 7 ได้รับการเอ็กซีคิว RST 7 นี้เป็นคำสั่งหนึ่งในไมโครโปรเซสเซอร์ 8080, 8085 และ Z80 ที่ใช้ให้ไปทำงานตำแหน่งที่ต้องการเมื่อเกิดการอินเทอรัพท์ซึ่งข้อความนี้จะถูกเปลี่ยน



รูปที่ 8.2 แสดงคำสั่งที่ใช้ใน DDT

เป็นตำแหน่งจุดเบรค (Break point) ใน DDT ดังนั้นเมื่อต้องการจุดเบรคในโปรแกรมก็ทำได้โดยการใช้คำสั่ง RST 7 ในตำแหน่งที่ต้องการ

เมื่อต้องการใช้ DDT และคีย์คำสั่ง DDT แล้ว DDT จะรอรับคำสั่งโดยแสดง " - " ซึ่งคำสั่งที่ใช้อนุญาตให้นำไฟล์จากดิสค์เข้ามาในหน่วยความจำ แสดงข้อมูลในหน่วยความจำหรือรีจิสเตอร์ของซีพียู ทำการแอสเซมเบลอร์และดิสแอสเซมเบลอร์โปรแกรมหรือตรวจสอบการทำงานทีละสเต็บเพื่อดูผลการทำงานของโปรแกรม

คำสั่งของ DDT

เราสามารถเลือกการทำงานคำสั่งต่าง ๆ ได้ดังแสดงดังรูปที่ 8.2 และเมื่อต้องการกลับไป CP/M จะทำได้ด้วยการวอร์มบูทคือคำสั่ง G0 หรือ CTRL-C นั่นเอง

การนำไฟล์ลงสู่หน่วยความจำ

หลังจาก DDT ได้รับการนำลงสู่หน่วยความจำแล้ว ก็จะต้องนำโปรแกรมที่ต้องการทดสอบปรับปรุงแก้ไขลงสู่หน่วยความจำด้วยและสามารถทำได้ 2 วิธี วิธีแรก โดยการพิมพ์ชื่อไฟล์ที่ต้องการตามหลังคำสั่ง DDT ดังตัวอย่าง

```
A>DDT FILEASM.HEX
DDT VERS 2.2
NEXT  PC
802D 0000
-
```

กรณีนี้จะนำไฟล์ FILEASM.HEX ลงในหน่วยความจำก่อนจะแสดงเครื่องหมาย " - " เพื่อรับคำสั่งถึงแม้ว่าตามปกติ FILEASM.HEX จะได้รับการเปลี่ยนจากไฟล์เลขฐาน 16 เป็นรหัสเลขฐานสอง (ซึ่งสามารถทำงานได้โดยคอมพิวเตอร์) DDT จะทำการเปลี่ยนเลขฐาน 16 เป็นเลขฐาน 2 โดยอัตโนมัติเมื่ออ่าน HEX ไฟล์ลงสู่หน่วยความจำ

วิธีที่สองโดยการใช้คำสั่ง I (Input) และ R (READ) ของ DDT

```
A>DDT
DDT VERS 2.2
-IFILEASM.HEX
-R
NEXT  PC
802D 0000
```

คำสั่ง I จะนำชื่อของไฟล์มาเก็บที่ไฟล์คอนโทรลบล็อก (FCB-File control block) (ดูภาคผนวก ก.) ซึ่งใช้เก็บตัวพารามิเตอร์ในหน่วยความจำและใช้ควบคุมการอ่านและเขียนดิสค์ภายใต้ CP/M ดังตัวอย่างเช่น

- IFILEASM.HEX

ชื่อ FILEASM.HEX จะได้รับการเก็บใน FCB จนกว่าจะได้รับการแทนที่ด้วยชื่อไฟล์ชื่ออื่นและไฟล์ต้องอยู่ในดิสค์ไดร์ฟที่กำกับอยู่

คำสั่ง R จะทำการอ่าน .COM หรือ .HEX ไฟล์จาก FCB ลงมาในหน่วยความจำ โปรดจำไว้เสมอว่าถ้าชื่อไฟล์ใน FCB ยังมีค่าคงเดิมการใช้คำสั่ง R จะนำไฟล์ที่ปรากฏชื่อใน FCB ลงมาในหน่วยความจำโดยไม่ต้องใช้คำสั่ง I อีกเลย

การแสดงและแก้ไขข้อความในโปรแกรม

DDT จะมีคำสั่งหลายคำสั่งใช้ในการแสดงค่าในหน่วยความจำซึ่งคำสั่งเหล่านี้สามารถตรวจสอบค่าทั้งในหน่วยความจำและค่ารีจีสเตอร์ของ CPU ขณะนั้นและจะเปลี่ยนหรือใส่ค่าในหน่วยความจำด้วยค่าที่เราต้องการ ดังนั้นจึงสามารถเปลี่ยนข้อมูล คำสั่งในโปรแกรมและค่าในรีจีสเตอร์ของCPU ขณะนั้น

คำสั่งแสดงผล คำสั่งแรกใช้ตรวจสอบค่าในหน่วยความจำตำแหน่งใดก็ได้โดยแสดงบรรทัดละ 16 ไบท์ โดยมีรูปแบบการใช้ดังนี้

-D s,f

s และ f คือ ค่าหน่วยความจำเริ่มต้นและสุดท้ายอาจไม่ใช้ก็ได้ ในกรณีที่ไม่ใช้ s และ f จะเริ่มต้นที่ 0100H และจะแสดงผล 16 บรรทัด ๆ ละ 16 ค่า โดยมีรูปแบบการแสดงค่าดังนี้

aaaa bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb bb cccccccccccccc

aaaa คือ ตำแหน่งแอดเดรสที่แสดงด้วยเลขฐาน 16 bb คือ ข้อมูล 16 ตัวแสดงด้วยเลขฐาน 16 cccccccccccccccc คือ รหัส ASCII แทนแต่ละค่าของ 16 ไบท์ ถ้าค่าไม่ใช่รหัส ASCII จะแสดงด้วย " . " แทน ตัวอย่างค่าของ 06H จะเป็นรหัส ASCII ของ CTRL-G ซึ่ง DDT จะแทนด้วย " . " ทั้งนี้เนื่องจาก CTRL-G ไม่ใช่รหัสของสัญลักษณ์ตัวพิมพ์และ 6FH เป็นค่า ASCII ของ " O " ดังนั้น DDT จะแสดงค่า " O "

ประโยชน์ของการแสดงค่า ASCII ก็เพื่อที่จะตรวจสอบส่วนที่เป็นข้อความ (message) ซึ่งอยู่ในโปรแกรมในรูปของรหัส ASCII

ถ้าเราใช้คำสั่ง D แล้วตามด้วยตำแหน่งเริ่มต้น (S) DDT จะแสดงข้อมูลเริ่มต้นที่ตำแหน่งนี้แทนที่จะเป็น 0100H และจะแสดง 16 บรรทัดและถ้าตำแหน่งสุดท้ายได้รับการกำหนดด้วย (f) DDT จะแสดงค่าในหน่วยความจำตั้งแต่ตำแหน่งเริ่มต้นถึงตำแหน่งสุดท้ายตามที่ระบุ ซึ่งจะไม่แสดงทีละ 16 บรรทัด แต่จะแสดงข้อมูลต่อเนื่องบนจอไปเรื่อย ๆ จนถึงตำแหน่งสุดท้าย ดังนั้นอาจใช้ CTRL-S เพื่อหยุดดูและ CTRL-S เพื่อให้แสดงค่าต่อไป

ถ้าต้องการแสดงรหัสโปรแกรม FILEASM ซึ่งผ่านการแปลจากบทที่แล้วเราสามารถนำลงหน่วยความจำโดยใช้คำสั่ง I และ R เนื่องจากมันได้รับการกำหนดให้เริ่มต้นที่ 8000H ดังนั้นคำสั่งแสดงผลจึงต้องเริ่มที่ตำแหน่งนี้ตามตัวอย่าง

```
-D8000
8000 C3 04 80 64 E5 D5 C5 F5 3E 00 D3 A1 3A 03 80 4F ...d....>...:..O
8010 06 64 3E 01 D3 A1 05 00 00 00 00 C2 16 80 06 64 .d>............d
8020 0D C2 16 80 3E 00 D3 A1 F1 C1 D1 E1 C9 01 00 00 ....>...........
8030 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ................
8040 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ................
8050 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ................
8060 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ................
8070 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ................
8080 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ................
8090 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ................
80A0 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ................
80B0 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ................
```

คำสั่งแสดงผลซึ่งจะแสดงค่าในหน่วยความจำทีละค่าในแต่ละครั้งและอนุญาตให้เปลี่ยนค่าในหน่วยความจำนั้นโดยใช้คำสั่งดังนี้

Sa

โดย a คือตำแหน่งแอดเดรสของหน่วยความจำที่ต้องการให้แสดงเมื่อกด<cr>ตำแหน่งแอดเดรสและค่าในหน่วยความจำนั้นจะแสดงออกมาทางจอ ถ้าต้องการเปลี่ยนค่าก็พิมพ์ค่าที่ต้องการเปลี่ยนลงไปแล้วตามด้วย <cr> ตำแหน่งแอดเดรสและค่าในตำแหน่งแอดเดรสของหน่วยความจำต่อมาจะแสดงทางจอและถ้าเราไม่ต้องการเปลี่ยนค่าในตำแหน่งนั้น ก็กด <cr>และถ้าต้องการออกจากคำสั่งนี้ทำได้โดยการพิมพ์ " . " แล้วตามด้วย <cr> ดังตัวอย่างข้างล่างนี้

```
A>DDT FILEASM.HEX
DDT VERS 2.2
NEXT  PC
802D 0000
-S8000
8000 C3
8001 04
8002 80
8003 64 0A
8004 E5 .
_
_
```

หากต้องการตรวจสอบค่าในรีจิสเตอร์ของซีพียูว่ามีค่าเป็นอย่างไร DDT มีคำสั่งในการตรวจสอบค่าคือ X โดยที่ X เป็นคำสั่ง ตรวจสอบค่าและเปลี่ยนแปลงค่าในรีจิสเตอร์ของไมโครโปรเซสเซอร์ 8080, 8085 เช่น โปรแกรมเคาวน์เตอร์ สแตคพอยท์เตอร์ แฟลกเงื่อนไขต่าง ๆ รูปแบบการใช้คำสั่ง

-X

หรือ

-Xr

โดย r คืออักษรบอกรีจิสเตอร์ของ 8080 ถ้ายกเว้นจะแสดงรีจิสเตอร์ทั้งหมด

CfZfMfEfIF A = bb B = dddd C = dddd H = dddd S = dddd P = dddd Inst

อักษรตัวพิมพ์ใหญ่ คือ ชื่อของรีจีสเตอร์ แฟลก หรือโปรแกรมเคาวน์เตอร์ f เป็น 0 หรือ 1 แทนค่าของแต่ละบิทของแฟลก bb คือค่าของไบท์เดี่ยวและ dddd คือค่าของสองไบท์และ inst คือ สัญลักษณ์คำสั่งชั่วขณะนั้นในหน่วยความจำที่อยู่ในโปรแกรมเคาวน์เตอร์

ดังนั้นถ้าต้องการทราบค่าปัจจุบันของ แอคคิวมูเลเตอร์ เราก็ทำได้โดยการใช้คำสั่ง

-X

DDT จะตอบสนองโดยการแสดง

```
-X
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=0000 JMP  DA03
-
```

รูปแบบการใช้คำสั่ง X แบบที่สองใช้แสดงค่าและเปลี่ยนแปลงค่าในรีจีสเตอร์ แฟลกหรือโปรแกรมเคาวน์เตอร์ตัวอย่างเช่น ถ้าเราต้องการเปลี่ยนค่าในแอคคิวมูเลเตอร์จาก 00H เป็น FFH เราต้องใช้คำสั่งดังนี้

```
-
-
-
-XA
A=00 FF
-XA
A=FF
-
-XB
B=0000 FFAA
-XB
B=FFAA
-
-
```

เมื่อใช้คำสั่ง Xr DDT จะแสดงค่าในแฟลก รีจีสเตอร์หรือโปรแกรมเคาวน์เตอร์และรอรับค่าใหม่ถ้าเราตอบรับโดยการกด <cr> ค่าจะไม่เปลี่ยนแปลง อย่างไรก็ตามถ้าเราพิมพ์ค่าใหม่ DDT จะเปลี่ยนค่าแฟลก รีจีสเตอร์ โปรมแกรมเคาวน์เตอร์ด้วยค่าใหม่

มีบ่อยครั้งที่เราต้องการตั้งค่าของกลุ่มของหน่วยความจำเป็นค่าบางค่าซึ่งสามารถทำได้ง่ายโดยการใช้คำสั่ง

F s,f,d

เมื่อ s คือ ตำแหน่งเริ่มต้นของหน่วยความจำและ f คือตำแหน่งสุดท้ายของหน่วยความจำและ d คือ ค่าข้อมูลในรูปของเลขฐาน 16 เพื่อที่จะแทนลงในตำแหน่งหน่วยความจำที่กำหนด ตัวอย่าง

```
A>DDT
DDT VERS 2.2
-D800,850
0800 41 20 58 52 41 20 4F 52 41 20 43 4D 50 20 49 4E A XRA ORA CMP IN
0810 52 20 44 43 52 20 4D 56 49 20 4C 58 49 20 53 54 R DCR MVI LXI ST
```

```
0820 41 58 49 4E 58 20 44 41 44 20 4C 44 41 58 44 43 AXINX DAD LDAXDC
0830 58 20 52 53 54 20 50 53 57 20 50 4F 50 20 50 55 X RST PSW POP PU
0840 53 48 4E 5A 5A 20 4E 43 43 20 50 4F 50 45 50 20 SHNZZ NCC POPEP
0850 4D M
-F800,850,E5
-D800,85F
0800 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ................
0810 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ................
0820 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ................
0830 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ................
0840 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ................
0850 E5 20 42 20 43 20 44 20 45 20 48 20 4C 20 4D 20 . B C D E H L M
-
-
```

หน่วยความจำตั้งแต่ 0800H ถึง 0850 จะได้รับการทำให้มีค่าเป็น E5

คำสั่ง M เป็นคำสั่งใช้ในการเคลื่อนย้าย บล็อกของหน่วยความจำจากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่งดังตัวอย่าง

M s,f,d

โดยที่ s คือตำแหน่งเริ่มต้นที่ต้องการย้าย f คือตำแหน่งสุดท้ายที่ต้องการย้ายข้อมูลและ d คือ ตำแหน่งเริ่มต้นของหน่วยความจำที่จะย้ายข้อมูลไปเก็บ

```
-
-DA000,A05F
A000 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ........
A010 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ........
A020 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ........
A030 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ........
A040 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ........
A050 01 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ........
-M0800,0850,A000
--DA000,A05F
A000 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ........
A010 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ........
A020 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ........
A030 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ........
A040 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 ........
A050 E5 01 00 00 01 01 00 00 01 01 00 00 01 01 00 00 ........
-
-
```

ซึ่งหมายความว่าย้ายข้อมูลหน่วยความจำจาก 0800 ถึง 0805 ไปไว้ยังจุดเริ่มต้นที่ A000

โปรแกรมภาษาแอสเซมบลี้

มีบ่อยครั้งขณะที่เรากำลังใช้งาน DDT เราอาจพบว่าโปรแกรมมีที่ผิดและต้องการที่จะแก้ข้อผิดในขณะนั้น DDT มีคำสั่ง A ซึ่งมีรูปแบบการใช้คำสั่งดังนี้ (คำสั่งนี้ทำให้สะดวกในการใช้งานเนื่องจากเราไม่ต้องออกจาก DDT เพื่อให้ ED แก้ไขโปรแกรมแล้วต้องใช้ ASM ใหมอีกด้วย)

Am

m คือตำแหน่งหน่วยความจำ เมื่อใช้คำสั่งนี้ DDT จะแสดงตำแหน่งในหน่วยความจำและรอให้ผู้ใช้พิมพ์คำสั่งด้วยภาษาแอสเซมบลี้ของไมโครโปรเซสเซอร์ 8080 DDT จะแปลเป็นรหัสภาษาเครื่องเก็บในหน่วยความจำแล้วพิมพ์ค่าหน่วยความจำถัดมาแล้วรอให้ผู้ใช้พิมพ์ คำสั่งภาษาแอสเซมบลี้ต่อไป ถ้าเรากด <cr>แทนที่จะพิมพ์คำสั่งภาษาแอสเซมบลี้จะเป็นการหยุดการทำงานของคำสั่ง A ตัวอย่างข้างล่างนี้จะแก้ไขค่าในหน่วยความจำ 0100H และจะดูค่าในหน่วยความจำเพื่อดูผลจากการใช้คำสั่ง สิ่งที่น่าสังเกตคือ Call 8000 เป็นคำสั่งต้องใช้หน่วยความจำ 3 ไบท์ ดังนั้นจะแสดง 103 ซึ่งเป็นตำแหน่งต่อไปถ้าเป็นคำสั่ง JMP ซึ่งเป็นคำสั่งใช้หน่วยความจำ 3 ไบท์ ตำแหน่งหน่วยความจำถัดมาจะเป็น 103

```
A>
A>DDT
DDT VERS 2.2
-D100,1FF
0100 01 10 10 C3 3D 01 43 4F 50 59 52 49 47 48 54 20  ....=.COPYRIGHT
0110 28 43 29 20 31 39 38 30 2C 20 44 49 47 49 54 41  (C) 1980, DIGITA
0120 4C 20 52 45 53 45 41 52 43 48 20 20 20 20 20 20  L RESEARCH
0130 44 44 54 20 56 45 52 53 20 32 2E 32 24 31 00 02  DDT VERS 2.2$1..
0140 C5 C5 11 30 01 0E 09 CD 05 00 3A BB F3 D6 03 CA  ...0......:.....
0150 58 01 3D CA 58 01 3E 01 3D 32 A4 08 2A EC 01 22  X.=.X.>.=2..*.."
0160 A6 08 3A EB 01 32 A5 08 C1 21 07 00 7E 3D 90 57  ..:..2...!..~=.W
0170 1E 00 D5 21 00 02 78 B1 CA 83 01 0B 7E 12 13 23  ...!..x.....~..#
0180 C3 76 01 D1 C1 E5 62 78 B1 CA A5 01 0B 7B E6 07  .v....bx.....{..
0190 C2 98 01 E3 7E 23 E3 6F 7D 17 6F D2 A1 01 1A 84  ....~#.o}.o.....
01A0 12 13 C3 87 01 D1 2E 00 E9 7F C9 21 A1 1D 70 2B  ...........!..p+
01B0 71 2A A0 1D 44 4D CD 9E 07 0E 3A CD 83 07 0E 20  q*..DM....:.....
01C0 CD 83 07 3A 7D 1D 32 A2 1D 3A 7E 1D 21 A2 1D BE  ...:}.2..:~.!...
01D0 DA F1 08 21 FC 1C 3A A2 1D BE D2 EA 08 2A A2 1D  ...!..:......*..
01E0 26 00 01 FD 1C 09 4E CD 83 07 21 FF 38 00 BB F3  &.....N...!.8...
01F0 08 21 FC 1C 36 00 01 3A 02 CD 4A 01 00 BB 12 14  .!..6..:..J.....
-
-A0100
0100  CALL 8000
0103
-D0100,01FF
0100 CD 00 80 C3 3D 01 43 4F 50 59 52 49 47 48 54 20  ....=.COPYRIGHT
0110 28 43 29 20 31 39 38 30 2C 20 44 49 47 49 54 41  (C) 1980, DIGITA
0120 4C 20 52 45 53 45 41 52 43 48 20 20 20 20 20 20  L RESEARCH
0130 44 44 54 20 56 45 52 53 20 32 2E 32 24 31 00 02  DDT VERS 2.2$1..
0140 C5 C5 11 30 01 0E 09 CD 05 00 3A BB F3 D6 03 CA  ...0......:.....
0150 58 01 3D CA 58 01 3E 01 3D 32 A4 08 2A EC 01 22  X.=.X.>.=2..*.."
0160 A6 08 3A EB 01 32 A5 08 C1 21 07 00 7E 3D 90 57  ..:..2...!..~=.W
0170 1E 00 D5 21 00 02 78 B1 CA 83 01 0B 7E 12 13 23  ...!..x.....~..#
0180 C3 76 01 D1 C1 E5 62 78 B1 CA A5 01 0B 7B E6 07  .v....bx.....{..
0190 C2 98 01 E3 7E 23 E3 6F 7D 17 6F D2 A1 01 1A 84  ....~#.o}.o.....
01A0 12 13 C3 87 01 D1 2E 00 E9 7F C9 21 A1 1D 70 2B  ...........!..p+
01B0 71 2A A0 1D 44 4D CD 9E 07 0E 3A CD 83 07 0E 20  q*..DM....:.....
01C0 CD 83 07 3A 7D 1D 32 A2 1D 3A 7E 1D 21 A2 1D BE  ...:}.2..:~.!...
01D0 DA F1 08 21 FC 1C 3A A2 1D BE D2 EA 08 2A A2 1D  ...!..:......*..
01E0 26 00 01 FD 1C 09 4E CD 83 07 21 FF 38 00 BB F3  &.....N...!.8...
01F0 08 21 FC 1C 36 00 01 3A 02 CD 4A 01 00 BB 12 14  .!..6..:..J.....
-
```

DDT จะทำการแปลคำสั่งภาษาแอสเซมบลี้เป็นภาษาเครื่องในลักษณะบรรทัดต่อบรรทัด ดังนั้นจึงไม่สามารถใช้คำสั่งเทียม (EQU, ORG ฯลฯ) หรือโอเปอแรนด์ (operand) ที่เป็นตัวแปรได้เช่น

```
-A0100
0100  CALL START
?
_
```

DDT จะพิมพ์เครื่องหมาย " ? " เป็นการบอกว่าผิด ซึ่งแสดงว่าในการเขียนโปรแกรมด้วยเอดิเตอร์เราจะกำหนดค่าหรือตัวแปรต่าง ๆ ได้ เช่น

SETBUFFER EQU 0D800H

แต่ในการใช้ใน DDT นั้นเราต้องใช้คำสั่งที่แท้จริงที่เป็นตัวเลขเพราะฉะนั้นจึงต้องใช้ลักษณะนี้

```
-A0100
0100  CALL 8000
0103
```

คำสั่ง L เป็นคำสั่งในส่วนแอสเซมเบลอร์และดิสแอสเซมเบลอร์ของ DDT คำสั่ง L ทำหน้าที่เรียกดูค่าในหน่วยความจำมาอยู่ในรูปคำสั่งภาษาแอสเซมบลี้ แทนที่จะเป็นตัวเลขฐาน 16 เหมือนคำสั่ง D มีรูปแบบการใช้คำสั่งดังนี้

L,s,f

s คือตำแหน่งแอดเดรสเริ่มต้นของหน่วยความจำและ f คือตำแหน่งแอดเดรสสุดท้ายของหน่วยความจำซึ่งทั้ง s และ f อาจเลือกหรือไม่เลือกใช้ก็ได้ ถ้าไม่มีทั้ง s และ f DDT จะนำค่าของโปรแกรมเคาวน์เตอร์ขณะนั้นเป็นตำแหน่งหน่วยความจำเริ่มต้นที่จะเรียกมาดูในรูปคำสั่งภาษาแอสเซมบลี้โดยทำการแสดง 12 บรรทัดในแต่ละครั้งที่ใช้คำสั่ง L อย่างไรก็ตาม DDT ส่วนแอสเซมเบลอร์และดิสแอสเซมเบลอร์สามารถได้รับการแทนที่ได้โดยโปรแกรมอื่นถ้าโปรแกรมที่นำเข้ามามีขนาดใหญ่ในกรณีนี้ DDT จะแสดงเครื่องหมาย " ? " เมื่อเราใช้คำสั่ง L ซึ่งถ้าเครื่องคอมพิวเตอร์ที่ใช้มีขนาดหน่วยความจำไม่ถึง 64K ก็ต้องแก้โดยการขยายหน่วยความจำหรือลดขนาดของโปรแกรมลง

```
-L0100
  0100  CALL  8000
  0103  JMP   013D
  0106  MOV   B,E
  0107  MOV   C,A
  0108  MOV   D,B
  0109  MOV   E,C
  010A  MOV   D,D
  010B  MOV   C,C
  010C  MOV   B,A
  010D  MOV   C,B
  010E  MOV   D,H
_
-L0100,0106
  0100  CALL  8000
  0103  JMP   013D
  0104  MOV   B,E
```

```
   0107
 -
 -L8000
   8000  JMP   8004
   8003  LDAX  B
   8004  PUSH  H
   8005  PUSH  D
   8006  PUSH  B
   8007  PUSH  PSW
   8008  MVI   A,00
   800A  OUT   A1
   800C  LDA   8003
   800F  MOV   C,A
   8010  MVI   B,64
```

การติดตามการทำงานของโปรแกรม

มีบ่อยครั้งขณะที่เราทำการปรับปรุงแก้ไขโปรแกรมอยู่นั้น อาจมีความจำเป็นต้องติดตามการทำงานทีละสเต็ปของโปรแกรมเพื่อที่จะดูว่าโปรแกรมทำงานตามที่เราต้องการหรือไม่หรืออาจเป็นการหาว่าทำไมโปรแกรมจึงไม่ทำงาน DDT สามารถทำงานนี้ได้โดยการใช้คำสั่ง

Tn

โดย n เป็นตัวเลขที่อาจเลือกใช้หรือไม่เลือกใช้ก็ได้ (อยู่ในรูปเลขฐาน 16) บอกจำนวนสเต็ปที่ต้องการให้ทำงาน ถ้า n ละไว้ DDT จะตีความว่า n เป็น 1 ซึ่งเป็นการทำงานสเต็ปเดียว ขณะที่ใช้คำสั่ง T นี้มีข้อความจำดังนี้

ข้อแรก การติดตามการทำงานโดยใช้คำสั่ง T จะใช้ จุดหยุดหรือจุดเบรค (break Point) โดยการใช้คำสั่ง RST7 ดังนั้นโปรแกรมที่นำเข้ามาตรวจสอบต้องไม่ใช้คำสั่งนี้หรือใช้คำสั่ง JUMP ไปยังตำแหน่งแอดเดรสของหน่วยความจำ 0038H ข้อสอง โปรแกรมที่ทำงานภายใต้คำสั่ง T จะทำงานช้าลงประมาณ 500 เท่า เช่น โปรแกรม FILEASM จะสร้างพัลส์ซึ่งมีความยาว 1 วินาที ที่ขา 0 ของพอร์ท A1H

ตัวอย่างข้างล่างนี้จะใช้โปรแกรม FILEASM ทำงาน 16 สเต็ปและตำแหน่งแอดเดรสของหน่วยความจำที่แสดงจะอยู่ในรูปเลขฐาน 16

```
-
-IFILEASM.HEX
-R
NEXT  PC
802D 0000
-XP
P=0000 0100
-T10
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=0100 CALL 8000
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00FE P=8000 JMP  8004
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00FE P=8004 PUSH H
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00FC P=8005 PUSH D
```

```
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00FA P=8006 PUSH B
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00F8 P=8007 PUSH PSW
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00F6 P=8008 MVI  A,00
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00F6 P=800A OUT  A1
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00F6 P=800C LDA  8003
COZOMOEOIO A=64 B=0000 D=0000 H=0000 S=00F6 P=800F MOV  C,A
COZOMOEOIO A=64 B=0064 D=0000 H=0000 S=00F6 P=8010 MVI  B,64
COZOMOEOIO A=64 B=6464 D=0000 H=0000 S=00F6 P=8012 MVI  A,01
COZOMOEOIO A=01 B=6464 D=0000 H=0000 S=00F6 P=8014 OUT  A1
COZOMOEOIO A=01 B=6464 D=0000 H=0000 S=00F6 P=8016 DCR  B
COZOMOEOIO A=01 B=6364 D=0000 H=0000 S=00F6 P=8017 NOP
COZOMOEOIO A=01 B=6364 D=0000 H=0000 S=00F6 P=8018 NOP  *8019
```

คำสั่ง U(Untrace Command) มีการทำงานเหมือนคำสั่ง T(Trace Command) เว้นแต่จะไม่แสดงสถานะของ CPU หลังการทำงานแต่ละสเต็ป เราใช้ในกรณีต้องการดูผลการทำงานของโปรแกรมยาว ๆ ในช่วงท้าย โดยการใช้คำสั่ง U ทำงานไปถึงส่วนท้ายที่ต้องการแล้วใช้คำสั่ง T ดูผลการทำงาน ในหลายกรณีเราอาจจะกำหนดค่าโปรมแกรมเคาวน์เตอร์เป็นตำแหน่งหน่วยความจำที่ต้องการจะตรวจสอบแล้วใช้คำสั่ง T อย่างไรก็ตามมีหลายโปรแกรมจะมีการกำหนดค่าเริ่มต้นของตัวแปร ดังนั้นการกระโดดข้ามส่วนของโปรแกรมที่กำหนดค่าเริ่มต้นจะทำให้ผลการทำงานของโปรแกรมที่ตรวจสอบไม่ถูกต้อง ดังนั้นคำสั่ง U จึงเป็นวิธีการที่เร็วที่จะไปถึงส่วนของโปรแกรมที่ต้องการภายใต้การควบคุมของ DDT

รูปแบบคำสั่ง Un

โดย n คือสเต็ปของการทำงานที่ต้องการและจะแสดงผลเหมือนคำสั่ง T ตรงบรรทัดแรกของโปรแกรมเท่านั้น ต่อจากนั้นก็จะให้โปรแกรมทำงานทีละสเต็ปตามจำนวนที่กำหนด n (n เป็นเลขฐาน 16) เมื่อทำงานเสร็จจะแสดงตำแหน่งของหน่วยความจำที่โปรแกรมจะทำงานต่อไป

ตัวอย่างการใช้งาน DDT

ในส่วนนี้เราจะอธิบายตัวอย่างการใช้งานของ DDT กับโปรแกรม FILEASM ผลที่ปรากฏบนจอมอนิเตอร์จะปรากฏดังที่แสดง

```
A>DDT
DDT VERS 2.2
-IFILEASM.HEX
-R
NEXT  PC
802D 0000
-XP
P=0000 8000
-S8003
8003 64 0A
8004 E5 .
-T15
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=8000 JMP  8004
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=8004 PUSH H
```

```
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00FE P=8005 PUSH D
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00FC P=8006 PUSH B
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00FA P=8007 PUSH PSW
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00F8 P=8008 MVI  A,00
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00F8 P=800A OUT  A1
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00F8 P=800C LDA  8003
COZOMOEOIO A=0A B=0000 D=0000 H=0000 S=00F8 P=800F MOV  C,A
COZOMOEOIO A=0A B=000A D=0000 H=0000 S=00F8 P=8010 MVI  B,64
COZOMOEOIO A=0A B=640A D=0000 H=0000 S=00F8 P=8012 MVI  A,01
COZOMOEOIO A=01 B=640A D=0000 H=0000 S=00F8 P=8014 OUT  A1
COZOMOEOIO A=01 B=640A D=0000 H=0000 S=00F8 P=8016 DCR  B
COZOMOEOIO A=01 B=630A D=0000 H=0000 S=00F8 P=8017 NOP
COZOMOEOIO A=01 B=630A D=0000 H=0000 S=00F8 P=8018 NOP
COZOMOEOIO A=01 B=630A D=0000 H=0000 S=00F8 P=8019 NOP
COZOMOEOIO A=01 B=630A D=0000 H=0000 S=00F8 P=801A NOP
COZOMOEOIO A=01 B=630A D=0000 H=0000 S=00F8 P=801B JNZ  8016
COZOMOEOIO A=01 B=630A D=0000 H=0000 S=00F8 P=8016 DCR  B
COZOMOEOIO A=01 B=620A D=0000 H=0000 S=00F8 P=8017 NOP
COZOMOEOIO A=01 B=620A D=0000 H=0000 S=00F8 P=8018 NOP  *8019
-A8017
8017  MOV    H,B
8018
-XP
P=8019 8000
-XS
S=00F8 100
-XA
A=01 00
-XB
B=620A 0000
-XD
D=0000
-T15
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=8000 JMP  8004
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=8004 PUSH H
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00FE P=8005 PUSH D
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00FC P=8006 PUSH B
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00FA P=8007 PUSH PSW
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00F8 P=8008 MVI  A,00
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00F8 P=800A OUT  A1
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=00F8 P=800C LDA  8003
COZOMOEOIO A=0A B=0000 D=0000 H=0000 S=00F8 P=800F MOV  C,A
COZOMOEOIO A=0A B=000A D=0000 H=0000 S=00F8 P=8010 MVI  B,64
COZOMOEOIO A=0A B=640A D=0000 H=0000 S=00F8 P=8012 MVI  A,01
COZOMOEOIO A=01 B=640A D=0000 H=0000 S=00F8 P=8014 OUT  A1
COZOMOEOIO A=01 B=640A D=0000 H=0000 S=00F8 P=8016 DCR  B
COZOMOEOIO A=01 B=630A D=0000 H=0000 S=00F8 P=8017 MOV  H,B
COZOMOEOIO A=01 B=630A D=0000 H=6300 S=00F8 P=8018 NOP
COZOMOEOIO A=01 B=630A D=0000 H=6300 S=00F8 P=8019 NOP
COZOMOEOIO A=01 B=630A D=0000 H=6300 S=00F8 P=801A NOP
COZOMOEOIO A=01 B=630A D=0000 H=6300 S=00F8 P=801B JNZ  8016
COZOMOEOIO A=01 B=630A D=0000 H=6300 S=00F8 P=8016 DCR  B
COZOMOEOIO A=01 B=620A D=0000 H=6300 S=00F8 P=8017 MOV  H,B
COZOMOEOIO A=01 B=620A D=0000 H=6200 S=00F8 P=8018 NOP  *8019
-
-
-XP
```

```
P=8019 0100
-XP
P=0100 8000
-U10
COZOMOEOIO A=01 B=620A D=0000 H=6200 S=00F8 P=8000 JMP  8004*801A
-
-XB
B=630A 0A0A
-T13
COZOMOEOIO A=01 B=0A0A D=0000 H=6300 S=00F0 P=801A NOP
COZOMOEOIO A=01 B=0A0A D=0000 H=6300 S=00F0 P=801B JNZ  8016
COZOMOEOIO A=01 B=0A0A D=0000 H=6300 S=00F0 P=8016 DCR  B
COZOMOEOIO A=01 B=090A D=0000 H=6300 S=00F0 P=8017 MOV  H,B
COZOMOEOIO A=01 B=090A D=0000 H=0900 S=00F0 P=8018 NOP
COZOMOEOIO A=01 B=090A D=0000 H=0900 S=00F0 P=8019 NOP
COZOMOEOIO A=01 B=090A D=0000 H=0900 S=00F0 P=801A NOP
COZOMOEOIO A=01 B=090A D=0000 H=0900 S=00F0 P=801B JNZ  8016
COZOMOEOIO A=01 B=090A D=0000 H=0900 S=00F0 P=8016 DCR  B
COZOMOEOIO A=01 B=080A D=0000 H=0900 S=00F0 P=8017 MOV  H,B
COZOMOEOIO A=01 B=080A D=0000 H=0800 S=00F0 P=8018 NOP
COZOMOEOIO A=01 B=080A D=0000 H=0800 S=00F0 P=8019 NOP
COZOMOEOIO A=01 B=080A D=0000 H=0800 S=00F0 P=801A NOP
COZOMOEOIO A=01 B=080A D=0000 H=0800 S=00F0 P=801B JNZ  8016
COZOMOEOIO A=01 B=080A D=0000 H=0800 S=00F0 P=8016 DCR  B
COZOMOEOIO A=01 B=070A D=0000 H=0800 S=00F0 P=8017 MOV  H,B
COZOMOEOIO A=01 B=070A D=0000 H=0700 S=00F0 P=8018 NOP
COZOMOEOIO A=01 B=070A D=0000 H=0700 S=00F0 P=8019 NOP
COZOMOEOIO A=01 B=070A D=0000 H=0700 S=00F0 P=801A NOP  *801B
-
-G0
```

เริ่มต้นโดยการนำ DDT ลงในหน่วยความจำแล้วนำ HEX ไฟล์ของ FILEASM ลงในหน่วยความจำด้วยคำสั่ง I (Input) และคำสั่ง R (Read) แล้วกำหนดค่าในโปรแกรมเคาวน์เตอร์เป็นค่าตำแหน่งหน่วยความจำเริ่มต้นของโปรแกรม FILEASM โดยคำสั่ง X สุดท้ายกำหนดค่าจำนวนวินาทีที่เราต้องการให้โปรแกรม FILEASM นับ เนื่องจากเราไม่ได้เรียกโปรแกรม FILEASM จากโปรแกรมภาษาเบสิคซึ่งจะกำหนดค่าการนับวินาทีนี้โดยใช้คำสั่ง POKE แต่ในที่นี้เราใช้คำสั่ง S

เมื่อมาถึงขั้นนี้เรากำหนดค่าเริ่มต้นเรียบร้อยจากการเริ่มต้นติดตามการทำงานของโปรแกรม FILEASM ตอนแรกนี้ 21 สเต็ป (เลขฐาน 16 เท่ากับ 15) ผ่านส่วนของโปรแกรมที่กำหนดค่าเริ่มต้นไปยังจุดส่วนของโปรแกรมทำการนับเวลา

อย่างไรก็ตามหากเราต้องการเปลี่ยนคำสั่งที่แอดเดรส 8017 ซึ่งแต่เดิมเป็น NOP มาเป็น MOV H,B ก็ทำได้โดยใช้คำสั่ง A8017 แล้วแก้ไข จากนั้นเซตค่าโปรแกรมเคาวน์เตอร์เป็น 8000 และ เซตค่าแสตคพอยท์เตอร์ เป็น 100 เซตค่ารีจีสเตอร์ A, B, C, D, E เป็น 00 หลังจากนั้นจะเทรซการทำงานดูใหม่ 21 สเต็ป จะเห็นว่าคำสั่งในแอดเดรส 8017 เป็น MOV H,B

หากต้องการเทรซโดยใช้คำสั่ง U ก็ทำได้เช่นกัน โดยเซตค่าโปรแกรมเคาวน์เตอร์เป็น 8000 แล้วใช้ U10 ผลที่เกิดขึ้นคือโปรแกรมจะเบรคที่ 801A หากต้องการเลิกการทำงานและกลับสู่ CP/M ทำได้โดยใช้ G0 หรือ กดคีย์ CTRL-C

การเก็บโปรแกรมลงดิสค์

เนื่องจากเราไม่ต้องการที่จะใช้ DDT ทุกครั้งที่เราต้องการทำให้โปรแกรมทำงานดังนั้นจึงมีคำสั่ง LOAD เพื่อเปลี่ยน HEX ไฟล์เป็น รหัสภาษาเครื่องและเก็บในรูปแบบ .COM ซึ่งมีรูปแบบการใช้ดังนี้

A>LOAD FILEASM

การทำงาน LOAD จะอ่าน FILEASM.HEX และเปลี่ยนเป็นรหัสไบนารีและเขียนกลับลงบนดิสค์เป็น FILEASM.COM โดย FILEASM.HEX ก็ยังคงอยู่บนดิสค์

วิธีที่ 2 ที่จะเก็บโปรแกรมลงบนดิสค์ โดยการใช้คำสั่ง SAVE ซึ่งเป็นคำสั่งของ CCP ซึ่งมีข้อจำกัดมากกว่าเพราะว่าเราต้องรู้ว่ามีกี่ บล็อก (256 ไบท์) โปรแกรมกินที่ในหน่วยความจำเท่าไรและโปรแกรมต้องเริ่มต้นที่ 0100H อย่างไรก็ตามในหลาย ๆ กรณีถ้ามีการปรับปรุงแก้ไขด้วย DDT (โดยการใช้คำสั่ง A) เป็นต้น เราก็ใช้คำสั่ง SAVE โดยไม่ต้องกลับไปแก้ไขไฟล์หลัก (.ASM ไฟล์) แล้วทำการแปลงเป็น .HEX ไฟล์ใหม่โดย ASM ซึ่งต้องใช้ LOAD เป็นรหัสไบนารี

9

ตัวอย่างการใช้ ED, ASM และ DDT

โปรแกรมจัดระบบงาน CP/M มียูทิลิตี้อยู่ 3 ตัวที่เป็นประโยชน์ต่อการพัฒนาโปรแกรมโดย ED ทำหน้าที่ช่วยในการเขียนเสมือนกับสร้างกระดาษ ปากกา ดินสอ ยางลบให้ ผู้ใช้จะเขียนอะไรก็เขียนลงบนกระดาษที่ ED จัดไว้ให้ได้ เมื่อเขียนหรือแก้ไขจนเป็นที่พอใจแล้วก็จะใช้ ASM ทำหน้าที่ในการแปลภาษาแอสเซมบลี้ที่เขียนขึ้นเป็นภาษาเครื่อง การแปลด้วย ASM นี้นับว่าสะดวกต่อผู้ใช้มาก เพราะผู้ใช้ไม่ต้องจดจำรหัสภาษาเครื่องเลยและเมื่อ ASM แปลให้แล้วเราก็ใช้ DDT ทำหน้าที่ตรวจสอบการทำงานและแก้ไขการทำงานเพื่อให้ได้ผลตามที่ต้องการ การตรวจสอบโดย DDT นับว่าสะดวกและทำให้รู้ถึงปัญหาของโปรแกรมได้เป็นอย่างดี

ดังนั้นผู้ใช้งานด้วยการเขียนด้วยภาษาแอสเซมบลี้จึงต้องรู้จักและใช้งานยูทิลิตี้ทั้งสามตัวนี้เป็นอย่างดี และเพื่อให้ศึกษาทำความเข้าใจเพิ่มเติมจึงขอยกตัวอย่างขึ้นทั้งสองตัวอย่าง

ตัวอย่างที่ 9.1

โปรแกรมที่เขียนขึ้นนี้ใช้สแกนหากลุ่มข้อมูลโดยการค้นหาข้อมูลตัวที่มากที่สุด โดยโปรแกรมเริ่มที่แอดเดรส 100 H เราเริ่มต้นด้วยการใช้ ED แล้วทำการ ASM จากนั้นจึงใช้ DDT

```
A>
A>ED SCAN.ASM        — เรียก ED เพื่อเปิดไฟล์ SCAN.ASM.

NEW FILE
    : *I
   1:  ;PROGRAM FOR DEMONSTRATING ED ASM AND DDT
   2:  ;                                           — ใช้ CTRL กับ I
   3: ^I          ORG^I   100H      ;START OF TRANSIENT AREA
   4:             MVI     B,LEN     ;LENGTH OF VECTOR TO SCAN
   5:             MVI     C,0       ;LARGEST VALUE SO FAR
   6:             LXI     H,VECT    ;BASE OF VECTOR
   7:  LOOP:      MOV     A,M       ;GET VALUE
   8:             SUB     C         ;LARGEST VALUE IN C?
   9:             JNZ     NFOUND    ;JUMP IF LARGER VALUE NOT FOUND
  10:                               ;NEW LARGEST VALUE, STORE IT TO C
  11:             MOV     C,A
  12:  NFOUND:    INX     H         ;TO NEXT ELEMENT     — สร้างโปรแกรมโดยส่วนที่ขีด
  13:             DCR     B         ;MORE TO SCAN?         เส้นใต้ หมายถึงการคีย์ผ่านเข้า
  14:             JNZ     LOOP      ;FOR ANOTHER           ทางคีย์บอร์ดส่วนที่ไม่ได้ขีดเส้น
  15:  ;
  16:  ;          END OF SCAN STORE C
```

ได้คือการตอบสนองของเครื่อง

```
  17:  ;
  18:          MOV     A,C       ;GET LARGEST VALUE
  19:          STA     LARGE
  20:          JMP     0         ;REBOOT
  21:  ;
  22:  ;       TEST DATA
  23:  VECT:   DB      2,0,4,3,5,6,1,5
  24:  LEN     EQU     $-VECT    ;LENGTH
  25:  LARGE:  DS      1         ;LARGEST VALUE ON EXIT
  26:          END
  27:  ^Z
```

— ใช้ CTRL-Z เพื่อออกสู่โหมดคำสั่ง

```
   :  *BOF
   1:  ;PROGRAM FOR DEMONSTRATING ED ASM AND DDT
   2:  ;
   3:          ORG     100H      ;START OF TRANSIENT AREA
   4:          MVI     B,LEN     ;LENGTH OF VECTOR TO SCAN
   5:          MVI     C,0       ;LARGEST VALUE SO FAR
   6:          LXI     H,VECT    ;BASE OF VECTOR
   7:  LOOP:   MOV     A,M       ;GET VALUE
   8:          SUB     C         ;LARGEST VALUE IN C?
   9:          JNZ     NFOUND    ;JUMP IF LARGER VALUE NOT FOUND
  10:                            ;NEW LARGEST VALUE, STORE IT TO C
  11:          MOV     C,A
  12:  NFOUND: INX     H         ;TO NEXT ELEMENT
  13:          DCR     B         ;MORE TO SCAN?
  14:          JNZ     LOOP      ;FOR ANOTHER
  15:  ;
  16:  ;       END OF SCAN STORE C
  17:  ;
  18:          MOV     A,C       ;GET LARGEST VALUE
  19:          STA     LARGE
  20:          JMP     0         ;REBOOT
  21:  ;
  22:  ;       TEST DATA
  23:  VECT:   DB      2,0,4,3,5,6,1,5
   1:  *E
```

— เลิกโปรแกรม ED ทำการเก็บไว้ในดิสค์โดยมีชื่อโปรแกรม SCAN.ASM

```
A>
A>ASM SCAN
CP/M ASSEMBLER - VER 2.0
0122
000H USE FACTOR
END OF ASSEMBLY
```

— เรียกโปรแกรม ASM. มาทำการแปลภาษาแอสเซมบลี้ของโปรแกรม SCAN.

```
A>
A>TYPE SCAN.PRN
```

— ทำการพิมพ์โปรแกรม SCAN.PRN เพื่อดูผลการแอสเซมเบลอร์

```
                        ;PROGRAM FOR DEMONSTRATING ED ASM AND DDT
                        ;
0100                            ORG     100H      ;START OF TRANSIENT AREA
0100 0608                       MVI     B,LEN     ;LENGTH OF VECTOR TO SCAN
0102 0E00                       MVI     C,0       ;LARGEST VALUE SO FAR
0104 211901                     LXI     H,VECT    ;BASE OF VECTOR
0107 7E                 LOOP:   MOV     A,M       ;GET VALUE
0108 91                         SUB     C         ;LARGEST VALUE IN C?
0109 C20D01                     JNZ     NFOUND    ;JUMP IF LARGER VALUE NOT FOUND
                                                  ;NEW LARGEST VALUE, STORE IT TO C
010C 4F                         MOV     C,A
010D 23                 NFOUND: INX     H         ;TO NEXT ELEMENT
010E 05                         DCR     B         ;MORE TO SCAN?
010F C20701                     JNZ     LOOP      ;FOR ANOTHER
                        ;
```

```
                        ;       END OF SCAN STORE C
                        ;
 0112 79                        MOV     A,C     ;GET LARGEST VALUE
 0113 322101                    STA     LARGE
 0116 C30000                    JMP     0       ;REBOOT
                        ;
                        ;       TEST DATA
 0119 0200040305VECT:           DB      2,0,4,3,5,6,1,5
 0008 =                 LEN     EQU     $-VECT  ;LENGTH
 0121                   LARGE:  DS      1       ;LARGEST VALUE ON EXIT
 0122                           END

A>
A>
A>DDT SCAN.HEX
```

— เรียก DDT เพื่อโหลดโปรแกรม SCAN.HEX มาทำการตรวจสอบ

```
DDT VERS 2.2
NEXT  PC
0121 0000
-X
```

— ดูรีจิสเตอร์ต่าง ๆ ในขณะนี้

```
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=0000 JMP  DA03
-
-
-XP
P=0000 100
```

— เซตค่ารีจิสเตอร์ P หรือโปรแกรมเคาวน์เตอร์ เป็น 100 H

```
-
-X
```

— ดูรีจิสเตอร์ต่าง ๆ

```
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=0100 MVI  B,08
-
-L100
```

— ตรวจดูแบบดิสแอสเซมเบลอร์ว่าโปรแกรมที่โหลดเข้ามาตรงหรือไม่

```
  0100  MVI   B,08
  0102  MVI   C,00
  0104  LXI   H,0119
  0107  MOV   A,M
  0108  SUB   C
  0109  JNZ   010D
  010C  MOV   C,A
  010D  INX   H
  010E  DCR   B
  010F  JNZ   0107
  0112  MOV   A,C
-L
```

— เรียกดิสแอสเซมเบลอร์ต่อไปอีก 12 บรรทัดสังเกตคำสั่ง JMP 0000

```
  0113  STA   0121
  0116  JMP   0000
  0119  STAX  B
  011A  NOP
  011B  INR   B
  011C  INX   B
  011D  DCR   B
  011E  MVI   B,01
  0120  DCR   B
  0121  ??=   20
  0122  MOV   D,D
-A116
```

— เปลี่ยนคำสั่ง JMP 000 เป็น RST7 เพื่อสร้างจุดเบรคของการดีบัก

```
0116  RST 7
0117  ___
```

— กดคีย์รีเทิร์น

```
-L113
```

— เรียกดิสแอสเซมเบลอร์ตั้งแต่แอดเดรส 113 สังเกตคำสั่ง RST

```
  0113  STA   0121
  0116  RST   07
  0117  NOP
  0118  NOP
  0119  STAX  B
  011A  NOP
  011B  INR   B
```

```
 011C  INX   B
 011D  DCR   B
 011E  MVI   B,01
 0120  DCR   B
-X                                                      — ดูค่ารีจิสเตอร์ต่าง ๆ อีกครั้ง
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=0100 MVI  B,08
-
-I                                                      — ทำการเทรซ 1 สเต็ป
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=0100 MVI  B,08*0102
-T
COZOMOEOIO A=00 B=0800 D=0000 H=0000 S=0100 P=0102 MVI  C,00*0104
-
-T
COZOMOEOIO A=00 B=0800 D=0000 H=0000 S=0100 P=0104 LXI  H,0119*0107
-
-T3                                                     — เทรซ 3 สเต็ป
COZOMOEOIO A=00 B=0800 D=0000 H=0119 S=0100 P=0107 MOV  A,M
COZOMOEOIO A=02 B=0800 D=0000 H=0119 S=0100 P=0108 SUB  C
COZOMOEOIO A=02 B=0800 D=0000 H=0119 S=0100 P=0109 JNZ  010D*010D
-
-D119                                  — ดูค่าในหน่วยความจำตั้งแต่แอดเดรส 119
0119 02 00 04 03 05 06 01 .......
0120 05 20 52 45 53 45 41 52 43 48 20 20 20 20 20 20 . RESEARCH
0130 44 44 54 20 56 45 52 53 20 32 2E 32 24 31 00 02 DDT VERS 2.2$1..
0140 C5 C5 11 30 01 0E 09 CD 05 00 3A BB F3 D6 03 CA ...0......:.....
0150 58 01 3D CA 58 01 3E 01 3D 32 A4 08 2A EC 01 22 X.=.X.>.=2..*.."
0160 A6 08 3A EB 01 32 A5 08 C1 21 07 00 7E 3D 90 57 ..:..2...!..ๅ=.ฟ
0170 1E 00 D5 21 00 02 78 B1 CA 83 01 0B 7E 12 13 23 ...!..x.....ๅ..#
0180 C3 76 01 D1 C1 E5 62 78 B1 CA A5 01 0B 7B E6 07 .v....bx.....{..
0190 C2 98 01 E3 7E 23 E3 6F 7D 17 6F D2 A1 01 1A 84 ....ๅ#.o}.o.....
01A0 12 13 C3 87 01 D1 2E 00 E9 7F C9 21 A1 1D 70 2B ............!..p+
01B0 71 2A A0 1D 44 4D CD 9E 07 0E 3A CD 83 07 0E 20 q*..DM....:.....
01C0 CD 83 07 3A 7D 1D 32 A2 1D 3A 7E 1D 21 A2 1D BE ...:}.2..:ๅ.!...
-
-X                                                      — สถานะของซีพียูขณะนี้
COZOMOEOIO A=02 B=0800 D=0000 H=0119 S=0100 P=010D INX  H
-
-T5                                           — ทำการเทรซโปรแกรมอีก 5 สเต็ป
COZOMOEOIO A=02 B=0800 D=0000 H=0119 S=0100 P=010D INX  H
COZOMOEOIO A=02 B=0800 D=0000 H=011A S=0100 P=010E DCR  B
COZOMOEOIO A=02 B=0700 D=0000 H=011A S=0100 P=010F JNZ  0107
COZOMOEOIO A=02 B=0700 D=0000 H=011A S=0100 P=0107 MOV  A,M
COZOMOEOIO A=00 B=0700 D=0000 H=011A S=0100 P=0108 SUB  C*0109 — จุดที่เบรค
-
-U5    — เทรซต่ออีก 5 สเต็ปโดยไม่ต้องแสดงผลระหว่างสเต็ป            คือ 109
COZ1MOEOIO A=00 B=0700 D=0000 H=011A S=0100 P=0109 JNZ  010D*0107
-
-X                                        — ผลของซีพียูเมื่อหลังการวิ่งไป 5 สเต็ป
COZOMOEOIO A=00 B=0600 D=0000 H=011B S=0100 P=0107 MOV  A,M
-
-G                                                      — ให้ซีพียูทำงานต่อไป
*0116                  — จุดเบรคที่ใส่ RST ไว้
-
-X                                              —สถานะของซีพียูเมื่อจบโปรแกรม
COZ1MOEOIO A=00 B=0000 D=0000 H=0121 S=0100 P=0116 RST  07
-
-XP                                           — เซตค่าโปรแกรมเคาวน์เตอร์เป็น 100H
P=0116 100
-
-X                                                      — ดูค่าในรีจิสเตอร์ต่าง ๆ
COZ1MOEOIO A=00 B=0000 D=0000 H=0121 S=0100 P=0100 MVI  B,08
-
-I10                                                    — เทรซการทำงานอีก 10 สเต็ป
```

```
ข้อมูลตัวแรก ─┐ ข้อมูลตัวที่ใหญ่ที่สุด

C0Z1M0E0I0 A=00 B=0000 D=0000 H=0121 S=0100 P=0100 MVI  B,08
C0Z1M0E0I0 A=00 B=0800 D=0000 H=0121 S=0100 P=0102 MVI  C,00
C0Z1M0E0I0 A=00 B=0800 D=0000 H=0121 S=0100 P=0104 LXI  H,0119
C0Z1M0E0I0 A=00 B=0800 D=0000 H=0119 S=0100 P=0107 MOV  A,M
C0Z1M0E0I0 A=02 B=0800 D=0000 H=0119 S=0100 P=0108 SUB  A^C  ← ในโปรแกรม
C0Z0M0E0I0 A=02 B=0800 D=0000 H=0119 S=0100 P=0109 JNZ  010D   การนำค่าจาก A
C0Z0M0E0I0 A=02 B=0800 D=0000 H=0119 S=0100 P=010D INX  H      ไปไว้ยัง C แต่
C0Z0M0E0I0 A=02 B=0800 D=0000 H=011A S=0100 P=010E DCR  B      โปรแกรมไม่ทำ
C0Z0M0E0I0 A=02 B=0700 D=0000 H=011A S=0100 P=010F JNZ  0107   นั่นคือ JNZ
C0Z0M0E0I0 A=02 B=0700 D=0000 H=011A S=0100 P=0107 MOV  A,M    ควรเป็น JC
C0Z0M0E0I0 A=00 B=0700 D=0000 H=011A S=0100 P=0108 SUB  C
C0Z1M0E0I0 A=00 B=0700 D=0000 H=011A S=0100 P=0109 JNZ  010D
C0Z1M0E0I0 A=00 B=0700 D=0000 H=011A S=0100 P=010C MOV  C,A
C0Z1M0E0I0 A=00 B=0700 D=0000 H=011A S=0100 P=010D INX  H
C0Z1M0E0I0 A=00 B=0700 D=0000 H=011B S=0100 P=010E DCR  B
C0Z0M0E0I0 A=00 B=0600 D=0000 H=011B S=0100 P=010F JNZ  0107*0107
-
-A109
0109 JC 10D
010C .
-G0                                   — จากการทำงานของ DDT

A>
A>
A>SAVE 1 SCAN.COM     — เก็บโปรแกรมที่แก้แล้วและใช้ชื่อเป็น SCAN.COM
A>DDT SCAN.COM                    — เรียก DDT กลับมาใหม่
DDT VERS 2.2
NEXT  PC
0200 0100
-
-L100                            — ดิสแอสเซมเบลอร์ออกมาดูใหม่
  0100  MVI  B,08
  0102  MVI  C,00
  0104  LXI  H,0119
  0107  MOV  A,M
  0108  SUB  C
  0109  JC   010D               — ส่วนของโปรแกรมที่แก้ไขแล้ว
  010C  MOV  C,A
  010D  INX  H
  010E  DCR  B
  010F  JNZ  0107
  0112  MOV  A,C
-XP
P=0100                   —เซตค่าโปรแกรมเคาวน์เตอร์เป็น 100
-
-T10                             — เทรซการทำงาน 10 สเต็ป
                   ┌ข้อมูลมาเก็บแล้ว (ถูกต้อง)

C0Z0M0E0I0 A=00 B=0000 D=0000 H=0000 S=0100 P=0100 MVI  B,08
C0Z0M0E0I0 A=00 B=0800 D=0000 H=0000 S=0100 P=0102 MVI  C,00
C0Z0M0E0I0 A=00 B=0800 D=0000 H=0000 S=0100 P=0104 LXI  H,0119
C0Z0M0E0I0 A=00 B=0800 D=0000 H=0119 S=0100 P=0107 MOV  A,M
C0Z0M0E0I0 A=02 B=0800 D=0000 H=0119 S=0100 P=0108 SUB  C
C0Z0M0E0I0 A=02 B=0800 D=0000 H=0119 S=0100 P=0109 JC   010D
C0Z0M0E0I0 A=02 B=0800 D=0000 H=0119 S=0100 P=010C MOV  C,A
C0Z0M0E0I0 A=02 B=0802 D=0000 H=0119 S=0100 P=010D INX  H
C0Z0M0E0I0 A=02 B=0802 D=0000 H=011A S=0100 P=010E DCR  B
C0Z0M0E0I0 A=02 B=0702 D=0000 H=011A S=0100 P=010F JNZ  0107
C0Z0M0E0I0 A=02 B=0702 D=0000 H=011A S=0100 P=0107 MOV  A,M
C0Z0M0E0I0 A=00 B=0702 D=0000 H=011A S=0100 P=0108 SUB  C
C1Z0M1E0I1 A=FE B=0702 D=0000 H=011A S=0100 P=0109 JC   010D
```

```
C1ZOM1EOI1 A=FE B=0702 D=0000 H=011A S=0100 P=010D INX  H
C1ZOM1EOI1 A=FE B=0702 D=0000 H=011B S=0100 P=010E DCR  B
C1ZOMOEOIO A=FE B=0602 D=0000 H=011B S=0100 P=010F JNZ  0107*0107
-
-X       — ดูว่ารีจิสเตอร์   — ค่าที่ถูกต้อง
C1ZOMOEOIO A=FE B=0602 D=0000 H=011B S=0100 P=0107 MOV  A,M
-
-G,108 — ทำงานต่อจนถึงแอดเดรส 108
*0108
-X                    — ข้อมูลตัวต่อไป
C1ZOMOEOIO A=04 B=0602 D=0000 H=011B S=0100 P=0108 SUB  C
-T
C1ZOMOEOIO A=04 B=0602 D=0000 H=011B S=0100 P=0108 SUB  C*0109
-T
COZOMOEOIO A=02 B=0602 D=0000 H=011B S=0100 P=0109 JC   010D*010C
-X
COZOMOEOIO A=02 B=0602 D=0000 H=011B S=0100 P=010C MOV  C,A
-G
*0116                                                  — ทำงานต่อจนจบ
-X
COZ1MOEOIO A=03 B=0003 D=0000 H=0121 S=0100 P=0116 RST  07
-
-S121                                                  — ดูค่าที่มากที่สุด
0121 03          -คิดผิด!
0122 52
0123 45
0124 53
0125 45
0126 41
0127 52
0128 43                                                —จบคำสั่ง S
-L100
  0100  MVI  B,08
  0102  MVI  C,00
  0104  LXI  H,0119
  0107  MOV  A,M
  0108  SUB  C
  0109  JC   010D
  010C  MOV  C,A
  010D  INX  H
  010E  DCR  B
  010F  JNZ  0107
  0112  MOV  A,C                  -ดูรหัสคำสั่ง
L
  0113  STA  0121
  0116  RST  07
  0117  NOP
  0118  NOP
  0119  STAX B
  011A  NOP
  011B  INR  B
  011C  INX  B
  011D  DCR  B
  011E  MVI  B,01
  0120  DCR  B
-XP
P=0116 100                              -รีเซตค่าโปรแกรมเคาวน์เตอร์ใหม่
-
-T                                       — ทำงานทีละสเต็ปเพื่อดูค่า
COZ1MOEOIO A=03 B=0003 D=0000 H=0121 S=0100 P=0100 MVI  B,08*0102
-T
                 เซตค่านับ
                    ↓
COZ1MOEOIO A=03 B=0803 D=0000 H=0121 S=0100 P=0102 MVI  C,00*0104
```

```
-T                                   ← เซตค่าที่ใหญ่ที่สุด
C0Z1M0E0I0 A=03 B=0800 D=0000 H=0121 S=0100 P=0104 LXI H,0119*0107
-T                                   ← แอดเดรสของข้อมูล
C0Z1M0E0I0 A=03 B=0800 D=0000 H=0119 S=0100 P=0107 MOV A,M*0108
-T                                   ← นำข้อมูลตัวแรกมายัง A
C0Z1M0E0I0 A=02 B=0800 D=0000 H=0119 S=0100 P=0108 SUB C*0109
-T
C0Z0M0E0I0 A=02 B=0800 D=0000 H=0119 S=0100 P=0109 JC  010D*010C
-T
C0Z0M0E0I0 A=02 B=0800 D=0000 H=0119 S=0100 P=010C MOV C,A*010D
-T                                   ← ข้อมูลย้ายมา C ถูกต้อง
C0Z0M0E0I0 A=02 B=0802 D=0000 H=0119 S=0100 P=010D INX H*010E
-T
C0Z0M0E0I0 A=02 B=0802 D=0000 H=011A S=0100 P=010E DCR B*010F
-T
C0Z0M0E0I0 A=02 B=0702 D=0000 H=011A S=0100 P=010F JNZ 0107*0107
-T
C0Z0M0E0I0 A=03 B=0702 D=0000 H=011A S=0100 P=0107 MOV A,M*0108
-T                                   ← ข้อมูลตัวที่สองมา A
C0Z0M0E0I0 A=00 B=0702 D=0000 H=011A S=0100 P=0108 SUB C*0109
-T                                   ← การลบทำให้ข้อมูลผิด
C1Z0M1E0I1 A=FE B=0702 D=0000 H=011A S=0100 P=0109 JC  010D*010D
-T
C1Z0M1E0I1 A=FE B=0702 D=0000 H=011A S=0100 P=010D INX H*010E
-
-L100
 0100  MVI  B,08
 0102  MVI  C,00
 0104  LXI  H,0119
 0107  MOV  A,M
 0108  SUB  C                        — คำสั่งนี้ควรเป็นคำสั่ง CMP
 0109  JC   010D
 010C  MOV  C,A
 010D  INX  H
 010E  DCR  B
 010F  JNZ  0107
 0112  MOV  A,C
-A108
0108 CMP C                           — แก้เป็น CMP
0109
-
-G0                                  — ออกไปยัง CP/M

A>
A>
A>SAVE 1 SCAN.COM                    — เก็บโปรแกรมไว้ยังดิสเก็ตต์ซึ่ง SCAN.COM
A>DDT SCAN.COM                       — เรียกกลับเข้า DDT ใหม่
DDT VERS 2.2
NEXT  PC
0200 0100
-
-XP                                  — ตรวจดูว่าโปรแกรมเคาวน์เตอร์
P=0100
-L116
  0116  RST  07
  0117  NOP
  0118  NOP
  0119  STAX B
  011A  NOP                          } ตรวจดูค่า RST
  011B  INR  B
  011C  INX  B
  011D  DCR  B
  011E  MVI  B,01
```

```
  0120  DCR   B }
  0121  INX   B }
-G,116
*0116                                                  — ทำงานจนถึงแอดเดรส 116
-XC                                                    — ดูค่าแฟลกตัวขาด
C1       — แฟลกตัวขาดเป็น 1
-X                                                     — ดูค่ารีจิสเตอร์ทุกตัว
C1Z1M0E0I0 A=06 B=0006 D=0000 H=0121 S=0100 P=0116 RST  07
-S121                                                  — ดูค่าคำตอบ
0121 06    — คำตอบถูกต้อง !
0122 52
0123 45
0124 53
-G0                                                    — ออกจาก DDT

A>
A>
A>ED SCAN.ASM                                          -เรียก ED มาแก้โปรแกรม
     : *NSUB ┌──────── กดคีย์ CTRL-Z
    8: *OLT  │
    8:       ↓     SUB      C       ;LARGEST VALUE IN C?
    8: *SSUB^ZCMP^ZOLT   ┌── แก้ไข
    8:             CMP      C       ;LARGEST VALUE IN C?
    8: *
    9:             JNZ      NFOUND  ;JUMP IF LARGER VALUE NOT FOUND
    9: *SNC^ZC^ZOLT

BREAK "#" AT ^Z
    9: *SNZ^ZC^ZOLT
    9:             JC       NFOUND  ;JUMP IF LARGER VALUE NOT FOUND
    9: *E          ↑
                   └──── แก้ไข
A>
A>ASM SCAN AAZ                                         — ทำการ ASM ใหม่
CP/M ASSEMBLER - VER 2.0
0122
000H USE FACTOR
END OF ASSEMBLY

A>
A>DDT SCAN.HEX                                         — เรียก DDT เพื่อทดสอบใหม่
DDT VERS 2.2
NEXT  PC
0121 0000                                              — ดูโปรแกรมตอนท้าย
-L116
  0116  JMP  0000
  0119  STAX B
  011A  NOP
  011B  INR  B
  011C  INX  B
  011D  DCR  B
  011E  MVI  B,01
  0120  DCR  B
  0121  ??=  20
  0122  MOV  D,D
  0123  MOV  B,L                                       — ให้ทำงานตั้งแต่ 100-116
-G100,116
*0116                                                  — ดูค่าในหน่วยความจำ
-D121 ┌──────── O.K.
0121 06 52 45 53 45 41 52 43 48 20 20 20 20 20 .RESEARCH
0130 44 44 54 20 56 45 52 53 20 32 2E 32 24 31 00 02 DDT VERS 2.2$1..
0140 C5 C5 11 30 01 0E 09 CD 05 00 3A BB F3 D6 03 CA ...0......:.....
0150 58 01 3D CA 58 01 3E 01 3D 32 A4 08 2A EC 01 22 X.=.X.>.=2..*.."
0160 A6 08 3A EB 01 32 A5 08 C1 21 07 00 7E 3D 90 57 ..:..2...!..~=.W
```

```
-G0                                                  — ออกจาก DDT

A>
A>SAVE 1 SCAN.COM                — เก็บโปรแกรมที่ถูกต้องไว้ใช้งาน
A>
A>
```

ตัวอย่างที่ 9.2

เป็นโปรแกรมจัดเรียงค่าตัวเลขโดยเขียนขึ้นด้วยภาษาแอสเซมบลี้ของ 8080 โปรแกรมนี้เขียนขึ้นด้วย ED เช่นกัน ลักษณะของโปรแกรมและคำอธิบายประกอบแสดงได้ดังต่อไปนี้

```
A>ED SORT.ASM                  — เปิดไฟล์ SORT.ASM เพื่อเขียนโปรแกรม

NEW FILE
      : *VI
        ;SORT PROGRAM IN CP/M ASSEMBLY LANGUAGE
        ;DATE MARCH 1,1984
        ;START AT THE BEGINNING OF THE TRANSIENT PROGRAM AREA
        ;
                ORG     100H
        SORT:   LXI     H,SW    ;ADDRESS SWITCH TOGGLE
                MVI     M,1     ;SET TO 1 FOR FIRST ITERATION
                LXI     H,I     ;ADDRESS INDEX
                MVI     M,0     ;I - 0
        ;
        ;COMPARE I WITH ARRAY SIZE
        COMP:   MOV     A,M     ;A REGISTER = I
                CPI     N-1     ;CY SET IF I < (N-1)
                JNC     CONT    ;CONTINUE IF I<=(N-2)
        ;
        ;END OF ONE PASS THROUGH DATA
                LXI     H,SW    ;CHECK FOR ZERO SWITCHES
                MOV  A,M! ORA A! JNZ SORT ;END OF SORT IF SW = 0
        ;
                RST     7
        ;GO TO RST 7 MEANS GOTO DEBUGGER INSTEAD OF REBOOT
        ;CONTINUE THIS PASS
        ;ADDRESSING I, SO LOAD AV(I) INTO REGISTERS
        ;
        CONT:   MOV  E,A! MVI  D,0! LXI  H,AV! DAD  D! DAD D
                MOV  C,M! MOV  A,C! INX  H! MOV B,M
        ;
        ;LOW ORDER BYTE IN A AND C, HIGH ORDER BYTE IN B
        ;
        ;MOV H AND L TO ADDRESS AV(I+1)
                INX     H
        ;
        ;COMPARE VALUE WITH REGS CONTAINING AV(I)
                SUB  M! MOV  D,A! MOV  A,B!  INX  H! SBB M  ;SUBTRACT
        ;
        ;BORROW SET IF AV(I+1)>AV(I)
                JC      INCI    ;SKIP IF IN PROPER ORDER
        ;
        ;CHECK FOR EQUAL VALUES
                ORA  D! JZ  INCI   ;SKIP IF AV(I)=AV(I+1)
                MOV  D,M! MOV  M,B! DCX  H!  MOV E,M
                MOV  M,C! DCX  H!  MOV M,D!  DCX H! MOV M,E
        ;
        ;INCREMENT SWITCH COUNT
                LXI     H,SW! INR  M
        ;
        ;INCREMENT I
```

```
INCI:   LXI     H,I! INR  M! JMP COMP
;
;DATA DEFINITION SECTION
;
SW:     DB      0          ;RESERVE SPACE FOR SWITCH COUNT
I:      DS      1          ;SPACE FOR INDEX
AV:     DW      5,100,30,50,20,7,1000,300,100,-32767
N       EQU     ($-AV)/2        ;COMPUTE N
        END

*B60T
```

สั่งพิมพ์โปรแกรมจากจุดเริ่มต้น 60 บรรทัดเพื่อตรวจสอบ

```
;SORT PROGRAM IN CP/M ASSEMBLY LANGUAGE
;DATE MARCH 1,1984
;START AT THE BEGINNING OF THE TRANSIENT PROGRAM AREA
;
        ORG     100H
SORT:   LXI     H,SW       ;ADDRESS SWITCH TOGGLE
        MVI     M,1        ;SET TO 1 FOR FIRST ITERATION
        LXI     H,I        ;ADDRESS INDEX
        MVI     M,0        ;I = 0
;
;COMPARE I WITH ARRAY SIZE
COMP:   MOV     A,M        ;A REGISTER = I
        CPI     N-1        ;CY SET IF I < (N-1)
        JNC     CONT       ;CONTINUE IF I<=(N-2)
;
;END OF ONE PASS THROUGH DATA
        LXI     H,SW       ;CHECK FOR ZERO SWITCHES
        MOV  A,M! ORA A! JNZ SORT  ;END OF SORT IF SW = 0
;
        RST     7
;GO TO RST 7 MEANS GOTO DEBUGGER INSTEAD OF REBOOT
;CONTINUE THIS PASS
;ADDRESSING I, SO LOAD AV(I) INTO REGISTERS
;
CONT:   MOV  E,A! MVI  D,0! LXI  H,AV! DAD  D! DAD D
        MOV  C,M! MOV  A,C! INX  H! MOV B,M
;
;LOW ORDER BYTE IN A AND C, HIGH ORDER BYTE IN B
;
;MOV H AND L TO ADDRESS AV(I+1)
        INX     H
;
;COMPARE VALUE WITH REGS CONTAINING AV(I)
        SUB  M! MOV  D,A! MOV  A,B!  INX  H! SBB M   ;SUBTRACT
;
;BORROW SET IF AV(I+1)>AV(I)
        JC      INCI       ;SKIP IF IN PROPER ORDER
;
;CHECK FOR EQUAL VALUES
        ORA  D! JZ  INCI   ;SKIP IF AV(I)=AV(I+1)
        MOV  D,M! MOV  M,B! DCX  H!  MOV E,M
        MOV  M,C! DCX  H!  MOV M,D!  DCX H! MOV M,E
;
;INCREMENT SWITCH COUNT
        LXI     H,SW! INR  M
;
;INCREMENT I
INCI:   LXI     H,I! INR  M! JMP COMP
;
;DATA DEFINITION SECTION
;
SW:     DB      0          ;RESERVE SPACE FOR SWITCH COUNT
I:      DS      1          ;SPACE FOR INDEX
AV:     DW      5,100,30,50,20,7,1000,300,100,-32767
N       EQU     ($-AV)/2        ;COMPUTE N
```

```
  56:           END
   1: *E
```

— จบส่วนของ ED เก็บข้อความทั้งหมดไว้ในดิสเก็ตต์

```
A>
A>ASM SORT
```

— เรียก ASM เพื่อทำการแอสเซมเบลอร์โปรแกรม SORT

```
CP/M ASSEMBLER - VER 2.0
015C
000H USE FACTOR
END OF ASSEMBLY
```

— การแอสเซมเบลอร์ไม่มีข้อผิดพลาด

ไฟล์สำหรับสำรอง — ซอร์สไฟล์ ไฟล์สำหรับพิมพ์ ไฟล์ภาษาเครื่อง

```
A>DIR SORT.*
A: SORT     BAK : SORT     ASM : SORT     PRN : SORT     HEX
A>STAT SORT.*
```

— ASM สร้างให้

```
 Recs  Bytes  Ext Acc
   12     2k    1 R/W A:SORT.ASM
    0     0k    1 R/W A:SORT.BAK
    3     1k    1 R/W A:SORT.HEX
   19     3k    1 R/W A:SORT.PRN
Bytes Remaining On A: 2k
```

— ดูสถานะจำนวน K ของแต่ละไฟล์

```
A>TYPE SORT.PRN
```

— สั่งพิมพ์โปรแกรมทั้งหมดออกมาดู

— ซอร์สโปรแกรม

แอดเดรส — รหัสภาษาเครื่อง

```
                   ;SORT PROGRAM IN CP/M ASSEMBLY LANGUAGE
                   ;DATE MARCH 1,1984
                   ;START AT THE BEGINNING OF THE TRANSIENT PROGRAM AREA
                   ;
0100                       ORG     100H
0100 214601        SORT:   LXI     H,SW       ;ADDRESS SWITCH TOGGLE
0103 3601                  MVI     M,1        ;SET TO 1 FOR FIRST ITERATION
0105 214701                LXI     H,I        ;ADDRESS INDEX
0108 3600                  MVI     M,0        ;I = 0
                   ;
                   ;COMPARE I WITH ARRAY SIZE
010A 7E            COMP:   MOV     A,M        ;A REGISTER = I
010B FE09                  CPI     N-1        ;CY SET IF I < (N-1)
010D D21901                JNC     CONT       ;CONTINUE IF I<=(N-2)
                   ;
                   ;END OF ONE PASS THROUGH DATA
0110 214601                LXI     H,SW         ;CHECK FOR ZERO SWITCHES
0113 7EB7C20001            MOV  A,M! ORA A! JNZ SORT  ;END OF SORT IF SW = 0
                   ;
0118 FF                    RST     7
                   ;GO TO RST 7 MEANS GOTO DEBUGGER INSTEAD OF REBOOT
                   ;CONTINUE THIS PASS
                   ;ADDRESSING I, SO LOAD AV(I) INTO REGISTERS
                   ;
0119 5F16002148CONT:       MOV  E,A! MVI  D,0! LXI  H,AV! DAD  D! DAD D
0121 4E792346              MOV  C,M! MOV  A,C! INX  H! MOV B,M
                   ;
                   ;LOW ORDER BYTE IN A AND C, HIGH ORDER BYTE IN B
                   ;
                   ;MOV H AND L TO ADDRESS AV(I+1)
0125 23                    INX     H
                   ;
                   ;COMPARE VALUE WITH REGS CONTAINING AV(I)
0126 9657782339E           SUB  M! MOV  D,A! MOV  A,B!  INX  H! SBB M  ;SUBTRACT
                   ;
                   ;BORROW SET IF AV(I+1)>AV(I)
012B DA3F01                JC      INCI     ;SKIP IF IN PROPER ORDER
                   ;
                   ;CHECK FOR EQUAL VALUES
012E B2CA3F01              ORA  D! JZ  INCI    ;SKIP IF AV(I)=AV(I+1)
0132 56702B5E              MOV  D,M! MOV  M,B! DCX  H!  MOV E,M
```

```
0136 712B722B73          MOV  M,C! DCX  H!  MOV M,D!  DCX H! MOV M,E
                     ;
                     ;INCREMENT SWITCH COUNT
013B 21460134            LXI     H,SW! INR  M
                     ;
                     ;INCREMENT I
013F 21470134C3INCI:     LXI     H,I! INR   M! JMP COMP
                     ;
                     ;DATA DEFINITION SECTION
                     ;
0146 00              SW:     DB     0           ;RESERVE SPACE FOR SWITCH COUNT
0147                 I:      DS     1           ;SPACE FOR INDEX
0148 050064001EAV:           DW     5,100,30,50,20,7,1000,300,100,-32767
000A =               N       EQU    ($-AV)/2        ;COMPUTE N
015C                         END
```

```
A>TYPE SORT.HEX                      — พิมพ์รหัสภาษาเครื่องในรูปฐานสิบหกออกมาดู
:10010000214601360121470136007EFE09D2190140
:100110002146017EB7C20001FF5F16002148011988
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:07014000470134C30A01006E
:100148000500640001E0032001400070E8032C01BB
:04015800640001800BE
:0000000000
```

```
A>
A>
A>DDT SORT.HEX     — เรียก DDT แล้วทำการโหลดโปรแกรม SORT.HEX เข้ามาเพื่อตรวจสอบ
DDT VERS 2.2
NEXT  PC
015C 0000
-XP                                                  — ตรวจดูค่าโปรแกรมเคาวน์เตอร์
P=0000 100 — เปลี่ยนค่า PC เป็น 100
-UFFFF                                    สถานะหลักการทำงาน   — ให้ทำงานไป 65535 สเต็ป
COZOMOEOIO A=00 B=0000 D=0000 H=0000 S=0100 P=0100 LXI  H,0146*0105
-
-XP                                                  -ดูค่าโปรแกรมเคาวน์เตอร์ใหม่
P=0105 100        — เซตเป็น 100 ใหม่
-T10                                                 — เทรซไปอีก 16 สเต็ป
COZOMOEOIO A=01 B=0000 D=0000 H=0146 S=0100 P=0100 LXI  H,0146
COZOMOEOIO A=01 B=0000 D=0000 H=0146 S=0100 P=0103 MVI  M,01
COZOMOEOIO A=01 B=0000 D=0000 H=0146 S=0100 P=0105 LXI  H,0147
COZOMOEOIO A=01 B=0000 D=0000 H=0147 S=0100 P=0108 MVI  M,00
COZOMOEOIO A=01 B=0000 D=0000 H=0147 S=0100 P=010A MOV  A,M  }
COZOMOEOIO A=00 B=0000 D=0000 H=0147 S=0100 P=010B CPI  09   }  — จุดที่สังเกต
C1ZOM1EOI1 A=00 B=0000 D=0000 H=0147 S=0100 P=010D JNC  0119 }
C1ZOM1EOI1 A=00 B=0000 D=0000 H=0147 S=0100 P=0110 LXI  H,0146
C1ZOM1EOI1 A=00 B=0000 D=0000 H=0146 S=0100 P=0113 MOV  A,M
C1ZOM1EOI1 A=01 B=0000 D=0000 H=0146 S=0100 P=0114 ORA  A
COZOMOEOIO A=01 B=0000 D=0000 H=0146 S=0100 P=0115 JNZ  0100
COZOMOEOIO A=01 B=0000 D=0000 H=0146 S=0100 P=0100 LXI  H,0146
COZOMOEOIO A=01 B=0000 D=0000 H=0146 S=0100 P=0103 MVI  M,01
COZOMOEOIO A=01 B=0000 D=0000 H=0146 S=0100 P=0105 LXI  H,0147
COZOMOEOIO A=01 B=0000 D=0000 H=0147 S=0100 P=0108 MVI  M,00
COZOMOEOIO A=01 B=0000 D=0000 H=0147 S=0100 P=010A MOV  A,M*010B
-A10D
010D  JC 119— เปลี่ยนตำแหน่งแอดเดรส 100 จาก JNC มาเป็น JC เพราะ
0110        — หลังจากตรวจสอบเฟรกแล้วเห็นว่ามันไม่กระโดดซึ่งผิดจุดประสงค์
-XP
P=010B 100  — เปลี่ยนค่าโปรแกรมเคาวน์เตอร์เป็น 100H.
```

```
-T10
C0Z0M0E0I0 A=00 B=0000 D=0000 H=0147 S=0100 P=0100 LXI  H,0146
C0Z0M0E0I0 A=00 B=0000 D=0000 H=0146 S=0100 P=0103 MVI  M,01
C0Z0M0E0I0 A=00 B=0000 D=0000 H=0146 S=0100 P=0105 LXI  H,0147
C0Z0M0E0I0 A=00 B=0000 D=0000 H=0147 S=0100 P=0108 MVI  M,00
C0Z0M0E0I0 A=00 B=0000 D=0000 H=0147 S=0100 P=010A MOV  A,M      — โปรแกรม
C0Z0M0E0I0 A=00 B=0000 D=0000 H=0147 S=0100 P=010B CPI  09         กระโดดไป
C1Z0M1E0I1 A=00 B=0000 D=0000 H=0147 S=0100 P=010D JC   0118
C1Z0M1E0I1 A=00 B=0000 D=0000 H=0147 S=0100 P=0119 MOV  E,A        แอดเดรส 119
C1Z0M1E0I1 A=00 B=0000 D=0000 H=0147 S=0100 P=011A MVI  D,00
C1Z0M1E0I1 A=00 B=0000 D=0000 H=0147 S=0100 P=011C LXI  H,0148
C1Z0M1E0I1 A=00 B=0000 D=0000 H=0148 S=0100 P=011F DAD  D
C0Z0M1E0I0 A=00 B=0000 D=0000 H=0148 S=0100 P=0120 DAD  D
C0Z0M1E0I0 A=00 B=0000 D=0000 H=0148 S=0100 P=0121 MOV  C,M
C0Z0M1E0I0 A=00 B=0005 D=0000 H=0148 S=0100 P=0122 MOV  A,C
C0Z0M1E0I0 A=05 B=0005 D=0000 H=0148 S=0100 P=0123 INX  H
C0Z0M1E0I0 A=05 B=0005 D=0000 H=0149 S=0100 P=0124 MOV  B,M*0125
-L100
  0100  LXI  H,0146                                          — ดิสแอสเซมเบลอร์ดู
  0103  MVI  M,01
  0105  LXI  H,0147
  0108  MVI  M,00
  010A  MOV  A,M
  010B  CPI  09
  010D  JC   0119                                            — ค่าที่แก้แล้วเป็น JC
  0110  LXI  H,0146
  0113  MOV  A,M
  0114  ORA  A
  0115  JNZ  0100
-L
  0118  RST  07
  0119  MOV  E,A
  011A  MVI  D,00
  011C  LXI  H,0148
  011F  DAD  D         —ดูเพิ่มเติมอีก
  0120  DAD  D
  0121  MOV  C,M
  0122  MOV  A,C
  0123  INX  H
  0124  MOV  B,M
  0125  INX  H
-G,127                                        — ทำงานจนถึงคำสั่งแอดเดรส 127
*0127
-T4                                           —เทรซการทำงาน 4 สเต็ป
C1Z0M1E0I0 A=A1 B=0005 D=0000 H=014A S=0100 P=0127 MOV  D,A
C1Z0M1E0I0 A=A1 B=0005 D=A100 H=014A S=0100 P=0128 MOV  A,B
C1Z0M1E0I0 A=00 B=0005 D=A100 H=014A S=0100 P=0129 INX  H
C1Z0M1E0I0 A=00 B=0005 D=A100 H=014B S=0100 P=012A SBB  M*012B
-D148
0148 05 00 64 00 1E 00 32 00 ..d...2.
0150 14 00 07 00 E8 03 2C 01 64 00 01 80 2A EC 01 22 ......,.d...*.."
0160 A6 08 3A EB 01 32 A5 08 C1 21 07 00 7E 3D 90 57 ..:..2...!..ๆ=.W
0170 1E 00 D5 21 00 02 78 B1 CA 83 01 0B 7E 12 13 23 ...!..x.....ๆ..#
0180 C3 76 01 D1 C1 E5 62 78 B1 CA A5 01 0B 7B E6 07 .v....bx.....{..
0190 C2 98 01 E3 7E 23 E3 6F 7D 17 6F D2 A1 01 1A 84 ....ๆ#.o}.o.....
01A0 12 13 C3 87 01 D1 2E 00 E9 7F C9 21 A1 1D 70 2B ...........!..p+
01B0 71 2A A0 1D 44 4D CD 9E 07 0E 3A CD 83 07 0E 20 q*..DM....:.....
01C0 CD 83 07 3A 7D 1D 32 A2 1D 3A 7E 1D 21 A2 1D BE ...:}.2..:ๆ.!...
01D0 DA F1 08 21 FC 1C 3A A2 1D BE D2 EA 08 2A A2 1D ...!..:......*..
01E0 26 00 01 FD 1C 09 4E CD 83 07 21 FF 38 00 BB F3 &.....N...!.8...
01F0 08 21 FC 1C 36 00 01 3A 02 CD 4A 01 00 BB 12 14 .!..6..:..J.....
-
-
-G0                                           — กลับไปยัง CP/M ใหม่
A>
A>DDT SORT.HEX                                — เรียกโปรแกรม SORT กลับเข้ามาใหม่
```

```
DDT VERS 2.2
NEXT  PC
015C 0000
-XP          }  — เซตค่าโปรแกรมเคาวน์เตอร์เป็น 100 ใหม่
P=0000 100   }
-L10D                                   —เรียกโปรแกรมจากแอดเดรส 10D มาดู
  010D  JNC  0119     — ส่วนที่ต้องแก้ไข
  0110  LXI  H,0146
  0113  MOV  A,M
  0114  ORA  A
  0115  JNZ  0100
  0118  RST  07
  0119  MOV  E,A
  011A  MVI  D,00
  011C  LXI  H,0148
  011F  DAD  D
  0120  DAD  D
-A10D
010D  JC 119          —แก้ไขให้ถูกต้อง
0110
-L100                                   — เรียกโปรแกรมออกมาดูใหม่
  0100  LXI  H,0146
  0103  MVI  M,01
  0105  LXI  H,0147
  0108  MVI  M,00
  010A  MOV  A,M
  010B  CPI  09
  010D  JC   0119                       — โปรแกรมส่วนที่แก้ไขแล้ว
  0110  LXI  H,0146
  0113  MOV  A,M
  0114  ORA  A
  0115  JNZ  0100
-A103                                   — เปลี่ยน "SWITCH" ด้วยค่าเริ่มต้นเป็น 00
0103  MVI M,0
0105
-^C                                     — กลับออกไปยัง CP/M
A>
A>
A>SAVE 1 SORT.COM                       — เก็บโปรแกรมไว้บนดิสเก็ตต์ โดยตั้งชื่อเป็น SORT.COM
A>
A>DDT SORT.COM                          — เรียกโปรแกรมมาดูใหม่ โดยใช้ DDT
DDT VERS 2.2
NEXT  PC
0200 0100                               — โปรแกรม COM จะได้ค่า PC เป็น 100H
-G          — ทำงานเมื่อ PC = 100H
*0118
-D148                   ┌──── โปรแกรมหยุดเมื่อพบ RST 7 ข้อมูลที่ได้รับการเก็บ
0148 05 00 07 00 14 00 1E 00 ........
0150 32 00 64 00 64 00 2C 01 E8 03 01 80 2A EC 01 22 2.d.d.,......*.."
0160 A6 08 3A EB 01 32 A5 08 C1 21 07 00 7E 3D 90 57 ..:..2...!..~=.W
0170 1E 00 D5 21 00 02 78 B1 CA 83 01 0B 7E 12 13 23 ...!..x.....~..#
0180 C3 76 01 D1 C1 E5 62 78 B1 CA A5 01 0B 7B E6 07 .v....bx.....{..
0190 C2 98 01 E3 7E 23 E3 6F 7D 17 6F D2 A1 01 1A 84 ....~#.o}.o.....
01A0 12 13 C3 87 01 D1 2E 00 E9 7F C9 21 A1 1D 70 2B .............!..p+
01B0 71 2A A0 1D 44 4D CD 9E 07 0E 3A CD 83 07 0E 20 q*..DM....:.....
01C0 CD 83 07 3A 7D 1D 32 A2 1D 3A 7E 1D 21 A2 1D BE ...:}.2..:~.!...
01D0 DA F1 08 21 FC 1C 3A A2 1D BE D2 EA 08 2A A2 1D ...!..:......*..
01E0 26 00 01 FD 1C 09 4E CD 83 07 21 FF 38 00 BB F3 &.....N...!.8...

  -G0                                   — กลับไปยัง CP/M ใหม่

  A>
  A>
  A>
  A>
  A>ED SORT.ASM        — เรียก ED มาเพื่อแก้ไขโปรแกรม
     : *N,0^ZOTT                        — หาข้อมูล 0 ตัวถัดไป
```

```
    9:              MVI     M,0      ;I - 0
    9: *-
    8:              LXI     H,I      ;ADDRESS INDEX
    8: *-
    7:              MVI     M,1      ;SET TO 1 FOR FIRST ITERATION
    7: *KT ลบทิ้ง
    7:              LXI     H,I      ;ADDRESS INDEX
    7: *I
    7:              MVI     M,0      ;ZERO SWITCH — เพิ่มเข้าไป
    8:
    8: *T
    8:              LXI     H,I      ;ADDRESS INDEX
    8: *NJNC^ZOT                                        — ค้นหา JNC
   14:              JNC*T
       CONT     ;CONTINUE IF I<=(N-2)
   14: *-2DIC^ZOLT                                      — ลบ N ออก

   14:              JC      CONT     ;CONTINUE IF I<=(N-2)
   14: *E

 -
  A>
  A>ASM SORT AAZ                 — เรียก ASM โปรแกรม SORT โดยไม่เอา .PRN ไฟล์
  CP/M ASSEMBLER - VER 2.0
  015C
  000H USE FACTOR
  END OF ASSEMBLY

A>
A>DDT SORT.HEX                   — เรียก DDT โหลดโปรแกรม SORT.HEX เข้ามา
DDT VERS 2.2
NEXT  PC
015C 0000
-G100                            — ทำโปรแกรมไปจนพบ RST 7
*0118
-D148                            — ข้อมูลที่ได้รับการ SORT แล้ว
0148 05 00 07 00 14 00 1E 00 ........
0150 32 00 64 00 64 00 2C 01 E8 03 01 80 2A EC 01 22 2.d.d.,.....*.."
0160 A6 08 3A EB 01 32 A5 08 C1 21 07 00 7E 3D 90 57 ..:..2...!..ๅ=.W
0170 1E 00 D5 21 00 02 78 B1 CA 83 01 0B 7E 12 13 23 ...!..x.....ๅ..#
0180 C3 76 01 D1 C1 E5 62 78 B1 CA A5 01 0B 7B E6 07 .v....bx.....{..
0190 C2 98 01 E3 7E 23 E3 6F 7D 17 6F D2 A1 01 1A 84 ....ๅ#.o}.o.....
01A0 12 13 C3 87 01 D1 2E 00 E9 7F C9 21 A1 1D 70 2B ...........!..p+
01B0 71 2A A0 1D 44 4D CD 9E 07 0E 3A CD 83 07 0E 20 q*..DM....:....
01C0 CD 83 07 3A 7D 1D 32 A2 1D 3A 7E 1D 21 A2 1D BE ...:}.2..:ๅ.!...
01D0 DA F1 08 21 FC 1C 3A A2 1D BE D2 EA 08 2A A2 1D ...!..:......*..
01E0 26 00 01 FD 1C 09 4E CD 83 07 21 FF 38 00 BB F3 &.....N...!.8...
01F0 08 21 FC 1C 36 00 01 3A 02 CD 4A 01 00 BB 12 14 .!..6..:..J.....
-G0 ———————————————————————————— ออกจาก DDT กลับเข้าสู่ CP/M
```

10

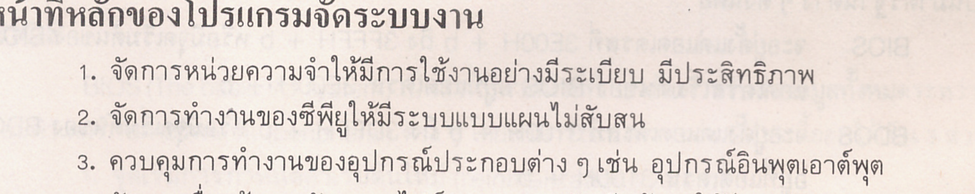
โครงสร้างโปรแกรม ระบบจัดการ CP/M

ในระบบจัดการของไมโครคอมพิวเตอร์ มีหลายรูปหลายระบบที่ใช้กันอยู่ เช่น CDOS ของบริษัทโครเมมโค, DOS 3.3 ของบริษัทแอปเปิ้ล ส่วนในระบบ 16 บิทก็มี MSDOS CP/M 86 XENIX, UNIX ฯลฯ โปรแกรมระบบจัดการของไมโครคอมพิวเตอร์จึงเปรียบเสมือนซอฟต์แวร์ที่จะเข้ามามีส่วนเกี่ยวข้องกับการจัดการทุกสิ่งทุกอย่างในระบบ เช่น รับคีย์บอร์ดว่ามีการกดคีย์อะไร เมื่อกดคีย์แล้วมีความหมายอย่างไร จะต้องทำอะไร แสดงผลบนจออย่างไร การทำงานในรูปแบบที่เกี่ยวกับฮาร์ดแวร์ของเครื่องนั้นล้วนแล้วแต่โปรแกรมจัดระบบงานเป็นตัวจัดการทั้งสิ้น

หน้าที่หลักของโปรแกรมจัดระบบงาน

1. จัดการหน่วยความจำให้มีการใช้งานอย่างมีระเบียบ มีประสิทธิภาพ
2. จัดการทำงานของซีพียูให้มีระบบแบบแผนไม่สับสน
3. ควบคุมการทำงานของอุปกรณ์ประกอบต่าง ๆ เช่น อุปกรณ์อินพุตเอาต์พุต
4. จัดการเรื่องข้อมูล จัดระบบไฟล์ ระบบการจัดเก็บข้อมูลให้มีประสิทธิภาพ

ลักษณะของโปรแกรมจัดระบบงานจึงเสมือนฐานรากของการใช้งาน โดยปกติเมื่อนำโปรแกรม CP/M มา จะเห็นว่ามีโปรแกรมอื่นติดมาด้วย ซึ่งโดยหลักการทำงานพื้นฐานแล้ว ผู้ใช้กับเครื่องจะมีโครงร่างจากฐานถึงยอด ดังนี้



รูปที่ 10.1 โครงสร้างของระบบซอฟต์แวร์ที่เชื่อมต่อกันเป็นชั้น ๆ

โดยปกติผู้ใช้มักจะรู้ในเรื่องโปรแกรมประยุกต์หรือใช้โปรแกรมประยุกต์ได้ แต่โปรแกรมเมอร์ก็รู้ตัวคอมพลายเลอร์และอินเตอร์พรีทเตอร์หรือภาษาคอมพิวเตอร์ ส่วนพวกที่จะทำการพัฒนาระบบจะต้องรู้เอดิเตอร์ แอสเซมเบลอร์ ดีบักเกอร์ ตลอดจนระบบโปรแกรมจัดระบบงานด้วยเพื่อจะได้เข้าใจการทำงานและวิธีการใช้งานอย่างมีประสิทธิภาพ

ส่วนประกอบของ CP/M ภายในโปรแกรมจัดระบบงานของ CP/M แบ่งออกได้เป็น 3 ส่วนดังต่อไปนี้ คือ

1. BIOS (the basic I/O system) เป็นส่วนที่เกี่ยวกับอินพุต-เอาต์พุต
2. BDOS (the basic disk operating system) ส่วนที่เกี่ยวกับจานแม่เหล็กและระบบไฟล์
3. CCP (the console command processor) ส่วนที่เกี่ยวกับการรับคำสั่งทางคีย์

และเมื่อ CP/M ได้รับโหลดเข้าสู่หน่วยความจำแล้วจะประกอบด้วยอีกสองส่วนคือ

4. TPA (the transient program area) เป็นส่วนที่นำโปรแกรมของผู้ใช้มาทำงาน
5. ส่วนพื้นที่เก็บพารามิเตอร์ของระบบ (System Parameter Area) เป็นส่วนสำหรับ CP/M ใช้งานเป็นกระดาษทด

ขนาดของหน่วยความจำของไมโครคอมพิวเตอร์ที่สามารถใช้ได้กับ CP/M จะต้องมีไม่น้อยกว่า 16 กิโลไบท์และไม่มากกว่า 64 กิโลไบท์ หน่วยความจำเหล่านี้จะต้องอยู่ในรูปจำนวนเต็มของ 1 กิโลไบท์ด้วยเพื่อว่าเมื่อ CP/M ได้รับการโหลดเข้าสู่หน่วยความจำของเครื่องไมโครคอมพิวเตอร์แล้ว จะได้อยู่ในตำแหน่งที่เป็นมาตรฐานต่าง ๆ ดังนี้คือ

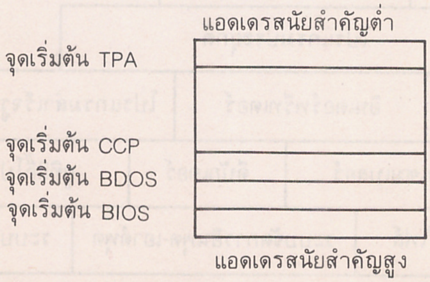
BIOS จะอยู่ตั้งแต่แอดเดรสที่ 3E00H + b ถึง 3FFFH + b หรือมีจุดเริ่มต้นของ BIOS แอดเดรสเริ่มต้นของ BIOS อยู่ที่แอดเดรส 3E00H + b

BDOS จะอยู่ตั้งแต่แอดเดรสที่ 3100H + b ถึง 3DFFH + b หรือมีจุดเริ่มต้นของ BDOS อยู่ที่แอดเดรส 3100H + b

CCP จะอยู่ตั้งแต่แอดเดรสที่ 2900H + b ถึง 30FFH + b หรือมีจุดเริ่มต้นของ CCP อยู่ที่แอดเดรส 2900 + b

TPA จะอยู่ตั้งแต่แอดเดรสที่ 100H ถึง 28FFH + b หรือมีจุดเริ่มต้นของ TPA อยู่ที่แอดเดรส 100H

และส่วนของพื้นที่เก็บพารามิเตอร์ของระบบจะเริ่มต้นตั้งแต่แอดเดรสที่ 00H ถึง 0FF H ลักษณะของตำแหน่งของจุดเริ่มต้นต่าง ๆ ได้แสดงไว้ในรูปที่ 9.1 ดังนี้



รูปที่ 10.2 แสดงลักษณะของโครงสร้างของส่วนต่าง ๆ ในหน่วยความจำ

ค่า " b " เป็นค่าไบแอส (bias) และจะขึ้นอยู่กับขนาดของหน่วยความจำและจะสามารถคำนวณหาได้จากสมการดังนี้ คือ

$$b = Mk - 16K$$

โดยที่ M เป็นขนาดของหน่วยความจำที่ต้องการจะคำนวณหาค่าไบแอส และต้องมีหน่วยเป็นกิโลไบท์เท่านั้น และค่า b ที่ได้จากสมการที่ 9.1 นี้จะมีค่าเป็นกิโลไบท์ด้วย ดังนั้นเมื่อจะนำค่า b มาใช้ต้องทำการเปลี่ยนเป็นตัวเลขฐาน 16 เสียก่อนดังตารางที่ 10.1

ขนาดของหน่วยความจำ	ค่าไบแอส	จุดเริ่มของ CCP	จุดเริ่มของ BDOS	จุดเริ่มของ BIOS
16K	16K – 16K = 00H = 00000H	02900H	03100H	03E00H
24K	24K – 16K = 08K = 02000H	04900H	05100H	05E00H
32K	32K – 16K = 16K = 04000H	06900H	07100H	07E00H
40K	40K – 16K = 24K = 06000H	08900H	09100H	09E00H
48K	48K – 16K = 32K = 08000H	0A900H	0B100H	0BE00H
56K	56K – 16K = 40K = 0A000H	0C900H	0D100H	0DE00H
62K	62K – 16K = 46K = 0B800H	0E100H	0E900H	0F600H
64K	64K – 16K = 48K = 0C000H	0E900H	0F100H	0FE00H

ตารางที่ 10.1 แสดงค่าจุดเริ่มต้นและค่าไบแอสของ CCP BDOS และ BIOS

BIOS

BIOS (The basic I/O system) เป็นส่วนที่ทำหน้าที่ที่เกี่ยวกับการรับส่งข้อมูลทั้งหมดระหว่างตัวคอมพิวเตอร์กับเพอริเฟอรัล (peripheral) ต่าง ๆ โดยสามารถแบ่งส่วนของ BIOS นี้ออกได้เป็น 3 ส่วนคือ

1. จุดเริ่มการกำหนดค่าเริ่มต้นใหม่ (reinitialization system)
2. ส่วนติดต่อกับอินพุตเอาต์พุตพื้นฐาน (Simple charactor I/O)
3. ส่วนติดต่อกับดิสเก็ตต์ (Diskette I/O)

1.จุดเริ่มการกำหนดค่าเริ่มต้นใหม่ (Reinitialization system) เป็นส่วนที่ทำการบูทระบบ ซึ่งในการบูทระบบนี้สามารถแบ่งออกได้เป็นสองแบบคือ โคลด์สตาร์ท (cold start) และวอร์มสตาร์ท (warm start)

วอร์มสตาร์ท (warm start) เป็นการสั่งให้ทำการโหลด CP/M จากส่วนของดิสเก็ตต์ในแทร็คที่ศูนย์และที่แทร็คที่ 1. จากแผ่นดิสค์ลงสู่หน่วยความจำตามตำแหน่งที่ได้ไบแอสไว้แล้ว หลังจากนั้นจะทำการจัดค่าจุดกระโดดต่าง ๆ ที่แอดเดรสที่ 0000 (ฐาน 16) ไปจนถึงแอดเดรสที่ 0007 (ฐาน 16) เพื่อเป็นจุดที่จะทำการเข้าสู่ส่วนของ BIOS และ BDOS ตามลำดับ นอกจากนี้ยังได้มีการจัดตารางของการกระโดด (jump table) ในส่วนของ BIOS เพื่อเข้าสู่ส่วนของโปรแกรมใน ROM และส่วนของตารางการกระโดดนี้จะเริ่มต้นตั้งแต่แอดเดรสที่ 3E00h + b เมื่อได้ทำการจัดการในตารางกระโดดนี้เรียบร้อยแล้วก็จะเข้าสู่ส่วนของ CCP เพื่อเตรียมรับคำสั่งจากแป้นพิมพ์ (CRT)

การทำวอร์มสตาร์ททำได้โดยการกดคีย์ control และคีย์ c โดยพร้อมกัน

โคลด์สตาร์ท (cold start) จะทำการจัดค่าภายในหน่วยความจำตั้งแต่แอดเดรสที่ 0040 (ฐาน 16) ไปถึงแอดเดรสที่ 004F (ฐาน 16) ให้มีค่าเป็นศูนย์ ต่อมาจะทำการจัดค่าใน IOBYTE (ที่แอดเดรสที่ 0003 ฐาน 16) ให้มีค่าเป็น 95 (ฐาน 16) การจัดค่าใน IOBYTE นี้ขึ้นกับตัวอุปกรณ์อินพุตเอาต์พุตที่ต่อไว้ ต่อมาจะทำการโปรแกรมการกำหนดค่าเริ่มต้นให้กับอุปกรณ์เพอริเฟอรัล และการตั้งอัตราการส่งข้อมูลระหว่างอุปกรณ์ที่เป็นอนุกรม เช่น CRT และเครื่องพิมพ์ หรือกำหนดสภาวะเริ่มต้นของอินพุตเอาต์พุตบางตัว ต่อจากนี้ก็จะไปกระทำการวอร์มสตาร์ท

การทำโคลด์สตาร์ททำได้โดยการกดปุ่มรีเซต (RESET) หรือปิดเปิดเครื่องใหม่

2. ส่วนติดต่อกับอินพุตเอาต์พุต (Simple charactor I/O) เป็นส่วนที่ทำการติดต่อกับเพอริเฟอรัลต่าง ๆ และทำการตรวจสอบสถานะต่าง ๆ ของคอนโซล (console) แบ่งออกได้เป็น 6 สถานะดังต่อไปนี้คือ

2.1 สถานะของคอนโซล (conslole status) จะทำการตรวจสอบสถานะของคอนโซล ว่าพร้อมที่จะรับหรือส่งข้อมูลหรือไม่ ถ้าพร้อมแล้ว ค่าในรีจิสเตอร์ A จะถูกจัดให้มีค่าเป็น 00 แต่ถ้ายังไม่พร้อมจะถูกจัดให้มีค่าเป็น 0FFH

2.2 คอนโซลอินพุต (console input) จะทำการอ่านข้อมูลจากคอนโซลซึ่งก็หมายถึงคีย์บอร์ดเข้าสู่รีจิสเตอร์ A

2.3 คอนโซลเอาต์พุต (console output) จะทำการส่งข้อมูลจากรีจิสเตอร์ C ออกทางคอนโซลเอาต์พุตซึ่งก็หมายถึงจอมอนิเตอร์

2.4 เครื่องพิมพ์ (List) ทำการส่งข้อมูลจากรีจิสเตอร์ C ออกทางหน่วยลิสต์ (list) (โดยปกติจะหมายถึงเครื่องพิมพ์)

2.5 เครื่องเจาะ (Punch) จะทำการส่งข้อมูลจากรีจิสเตอร์ C ออกสู่หน่วยเครื่องเจาะ (punch) (เครื่องเจาะเทปกระดาษ)

2.6 อุปกรณ์อ่านข้อมูล (Reader) ทำการอ่านข้อมูลจากหน่วยอุปกรณ์อ่านข้อมูล (เครื่องอ่านเทปกระดาษ) เข้าสู่รีจิสเตอร์ A

3. ส่วนติดต่อกับดิสเก็ตต์ (Diskette I/O) เป็นส่วนที่ติดต่อกับดิสเก็ตต์ทั้งหมด แบ่งออกได้เป็น 7 จุดด้วยกันดังนี้ คือ

3.1 โฮม (home) เป็นจุดที่จะเข้าสู่โปรแกรมที่จะสั่งให้หัวอ่านของดิสค์เคลื่อนที่กลับไปยังตำแหน่งที่แทร็คที่ศูนย์

3.2 เลือกดิสค์ไดร์ฟ (select disk drive) จะเป็นจุดที่จะเข้าสู่โปรแกรมที่จะทำการควบคุมให้การติดต่อกับดิสค์ (การเขียนหรือการอ่านกับดิสค์) ในครั้งต่อไปจะติดต่อยังไดร์ฟใด โดยค่าที่อยู่ภายในรีจิสเตอร์ C จะเป็นตัวบอกให้ทราบ คือ ถ้าค่าภายในรีจิสเตอร์ C มีค่าเป็น 0 1 2 หรือ 3 ค่าใดค่าหนึ่ง จะหมายถึงไดร์ฟ A, B, C และ D ตามลำดับ

3.3 กำหนดแทร็ค (set track) จะทำการจัดค่าของแทร็คตามค่าที่อยู่ภายในรีจิสเตอร์ C ค่าแทร็คที่ถูกจัดค่านี้จะเป็นชื่อแทร็คที่จะถูกทำการติดต่อ (เช่นการเขียนหรืออ่านกับดิสค์) กับดิสค์ในครั้งต่อไป ค่าที่ถูกจัดไว้ภายในรีจิสเตอร์ C นี้ จะต้องมีค่าอยู่ตั้งแต่ 0 ถึง 76 (0 – 4C ฐาน 16)

3.4 กำหนดเซกเตอร์ (set sector) จะทำการจัดค่าของเซกเตอร์ตามค่าที่อยู่ภายในรีจิสเตอร์ C ค่าเซกเตอร์ที่ถูกจัดค่าไว้นี้จะเป็นชื่อเซกเตอร์ที่จะถูกทำการติดต่อกับดิสค์ในการเขียนหรืออ่านในครั้งต่อไป

3.5 กำหนดแอดเดรสที่จะดีเอ็มเอ (set DMA address) จะทำการจัดค่าแอดเดรสเริ่มต้นของข้อมูลภายในหน่วยความจำที่จะถูกกระทำการ DMA (direct memory access) โดยค่าแอดเดรสนี้จะนำมาจากค่าที่ได้จัดไว้ภายในคู่รีจิสเตอร์ BC

3.6 การอ่าน (Read) จะทำการอ่านข้อมูลจากแผ่นดิสค์เป็นจำนวน 1 เซกเตอร์ ลงสู่หน่วยความจำในลักษณะที่คล้ายกับการกระทำ DMA โดยการอ่านนี้จะทำการอ่านข้อมูลจากแทร็คและเซกเตอร์ที่ได้ทำการจัดค่าไว้แล้วตามข้างบนลงสู่หน่วยความจำที่แอดเดรสตั้งแต่แอดเดรสเริ่มต้นของแอดเดรสของ DMA ที่ได้ถูกจัดค่าไว้แล้วตามหัวข้อที่ 5

3.7 การเขียน (Write) จะทำการเขียนข้อมูลจากหน่วยความจำจำนวนหนึ่งเซกเตอร์เข้าสู่แผ่นดิสค์ยังไดร์ฟ แทร็คและเซกเตอร์ ที่ได้กำหนดไว้แล้วจากคำสั่งก่อนหน้านี้

ส่วนของ BIOS ที่ใช้กับ CP/M Version 1.4A OS. (operating system) ที่ใช้อยู่กับเครื่องไมโครคอมพิวเตอร์อิมไซด์ของห้องปฏิบัติการวิจัยไมโครฯ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ จะอยู่ภายใน EPROM ที่ถูกจัดให้อยู่ตั้งแต่แอดเดรสที่ F000 (ฐานสิบหก) ถึงแอดเดรสที่ F531 (ฐานสิบหก) เมื่อทำการวอร์มสตาร์ท จะมีโปรแกรมส่วนหนึ่งภายใน EPROM (ที่แอดเดรสตั้งแต่ F10E ฐานสิบหก) ที่จะทำการจัดค่าภายในหน่วยความจำตั้งแต่แอดเดรสที่ 3E00 + b (ฐานสิบหก) เป็นต้นไปจำนวน 45 ไบท์ ให้เป็นส่วนของ BIOS ที่จะทำหน้าที่ในการกระโดดเข้าสู่ EPROM ต่อไป โดยคำสั่งทั้งหมดนี้และความหมายได้แสดงไว้ในตารางที่ 10.2ดังต่อไปนี้

ตารางที่ 10.2 แสดงส่วนของโปรแกรมของ BIOS

```
3E00H+b C31CF0  JMP    BOOT    ; ARRIVE HERE FROM COLD START LOAD!
3E03H+b C38CF0  JMP    WBOOT   ; ARRIVE HERE FOR WARM START!
3E06H+b C306F4  JMP    CONST   ; CHECK FOR CONSOLE CHAR  STATUS
3E09H+b C3DDF3  JMP    CONSIN  ; READ CONSOLE CHARACTER INPUT
3E0CH+b C30CF4  JMP    CONOUT  ; WRITE CONSOLE CHARACTER OUTPUT
3E0FH+b C315F4  JMP    LIST    ; WRITE LISTING CHARACTER OUTPUT
3E12H+b C312F4  JMP    PUNCH   ; WRITE CHARACTER TO PUNCH DEVICE
3E15H+b C30FF4  JMP    READER  ; READ  READER INPUT
3E18H+b C359F4  JMP    HOME    ; MOVE TO TRACK 00 ON SELECTED DISK
3E1BH+b C354F1  JMP    SELDSK  ; SELECT DISK DRIVE
3E1EH+b C397F1  JMP    SETTRK  ; SET TRACK NUMBER
3E21H+b C39CF1  JMP    SETSEC  ; SET SECTOR NUMBER
3E24H+b C3A1F1  JMP    SETDMA  ; SET DMA ADDRESS
3E27H+b C321F1  JMP    READER  ; READ SECTOR
3E2AH+b C344F2  JMP    WRITE   ; WRITE SECTOR
```

ที่แอดเดรส 3E30 + b (ฐานสิบหก) ต่อไปอีกเป็นจำนวน 4 ไบท์ จะเป็นคำสั่งดังต่อไปนี้ คือ

```
3E30 + b        EI
3E31 + b        JMP     2900H + b ; jmp to CCP
```

เมื่อทำการวอร์มสตาร์ทเรียบร้อยแล้ว ค่าภายในของโปรแกรมเคาน์เตอร์ (PC) จะได้รับการจัดให้เป็นแอดเดรส 3E30 + b (ฐานสิบหก) เพื่อทำการอีนาเบิ้ลอินเทอร์รัพท์ แล้วจะทำการกระโดดเข้าสู่ส่วนของ CCP ต่อไป

BDOS

BDOS (The basic disk operating system) เป็นโปรแกรมที่ใช้ในการจัดการเกี่ยวกับไฟล์ข้อมูลภายในแผ่นดิสค์ (disk file) ทั้งหมด ซึ่งสามารถแบ่งหน้าที่หลักออกได้เป็น 8 แบบด้วยกันดังนี้

— SEARCH ทำการค้นหาชื่อของไฟล์ข้อมูล (file name)
— OPEN ทำการเปิดการติดต่อกับไฟล์ข้อมูลเพื่อทำการอย่างใดอย่างหนึ่ง
— CLOSE ทำการปิดการติดต่อกับไฟล์ข้อมูล
— RENAME ทำการเปลี่ยนชื่อของไฟล์ข้อมูลใหม่
— READ ทำการอ่านข้อมูลภายในไฟล์ข้อมูลจากแผ่นดิสค์ลงสู่หน่วยความจำ
— WRITE ทำการเขียนข้อมูลจากหน่วยความจำกลับไปยังไฟล์ข้อมูลในแผ่นดิสค์
— SELECT ทำการเลือกดิสค์ไดร์ฟ สำหรับการกระทำต่อไป
— DELETE ทำการลบชื่อของไฟล์ข้อมูลออกจากส่วนของไดเรกทอรี (directory) หรือสารบัญ

1. SEARCH เป็นการค้นหาชื่อของไฟล์ข้อมูล โดย BDOS จะทำการอ่านชื่อของไฟล์ข้อมูลจากส่วนของสารบัญที่อยู่ภายในแทร็คที่ 2 กลุ่มที่ศูนย์และกลุ่มที่หนึ่ง (ดูรายละเอียดจากภาคผนวก ค.) มาเปรียบเทียบว่าชื่อของไฟล์ข้อมูลและชนิดของไฟล์ข้อมูล (file type) ตรงกันหรือไม่ เมื่อตรงกันจึงจะทำการเปิดไฟล์ข้อมูลนั้นต่อไป

2. OPEN BDOS จะทำการโหลดเอา FCB (File Control Block) (ดูรายละเอียดในภาคผนวก ค.) ของไฟล์ข้อมูลที่ต้องการจะทำการเปิดไปไว้ที่ TFCB ที่ตำแหน่งแอดเดรส 005C (ฐานสิบหก)

3. CLOSE จะทำการโหลดเอา TFCB ของไฟล์ข้อมูลนั้นลงไปยังส่วนที่เป็นสารบัญภายในแทร็คที่ 2

4. RENAME จะทำการเปลี่ยนชื่อของไฟล์ข้อมูล และ/หรือ ชนิดของไฟล์ข้อมูล โดยจะมีขั้นตอนดังนี้คือ Search, Open, Rename และ Close ตามลำดับ

5. READ จะทำการอ่านข้อมูลจากแผ่นดิสค์ตามกรุ๊ฟที่ระบุไว้ภายใน FCB

6. WRITE จะทำการเขียนข้อมูลลงในแผ่นดิสค์ พร้อมทั้งเขียนชื่อกรุ๊ฟที่แฟ้มข้อมูลนั้นถูกเขียนลงไปยังแผ่นดิสค์ลงบนส่วนของ FCB ด้วย

7. SELECT เป็นส่วนของโปรแกรมที่ใช้ตรวจสอบดูว่าในการติดต่อกับแผ่นดิสค์ในครั้งต่อไปจะทำการติดต่อกับดิสค์ไดร์ฟใด และเมื่อทำการติดต่อเรียบร้อยแล้ว ก็จะทำการควบคุมให้ส่งสัญลักษณ์ที่แสดงไดร์ฟ ของดิสค์ที่ติดต่อนั้นขึ้นบนคอนโซล (CRT)

8. DELETE เป็นการลบชื่อของไฟล์ข้อมูลออกจากดิสค์แผ่นนั้น โดย BDOS จะทำการหา FCB ของไฟล์ข้อมูลนั้นแล้วเขียนค่า E5 (ฐานสิบหก) ลงไปยังไบท์แรกของ FCB นั้น

ส่วนของ BDOS และ BIOS เมื่อรวมกันแล้วจะเรียกเป็นชื่อใหม่ว่า ‘‘FDOS’’

CCP

CCP (The Console Command processor) เป็นโปรแกรมที่ทำหน้าที่ในการติดต่อกับผู้ใช้งาน โดยจะรับคำสั่งจากผู้ใช้งาน (user) ทางคีย์บอร์ด

ในการทำงานของ CCP นี้ เมื่อได้รับคำสั่งที่เป็นสัญลักษณ์ ASCII จำนวน 1 ไบท์จากแป้นพิมพ์ แล้วจะทำการตัดบิท (bit) ที่ 7 (บิทที่มีนัยสำคัญสูงที่สุด) ทิ้ง แล้วจึงทำการตรวจสอบดูว่าเป็นคำสั่งของสัญลักษณ์ <Cr> หรือไม่ถ้าไม่ใช่ ก็จะทำการนำตัวสัญลักษณ์นั้นไปเก็บไว้ยังบัฟเฟอร์สแตค (stack area) ที่หนึ่งจนกว่าจะพบสัญลักษณ์ <Cr> (Carrage Return) หรือตัวอักษรควบคุม (Control Character) ต่าง ๆ ถึงจะนำข้อมูลในบัฟเฟอร์สแตค (stack area) นี้ไปทำการตีความ

คำสั่งต่าง ๆ ที่ใช้ใน CP/M จะแบ่งออกได้เป็น 2 ชนิดคือ

1. คำสั่งที่อยู่ภายใน CCP (Build-in Command)
2. คำสั่งที่อยู่ใน TPA หรือผู้ใช้สร้างขึ้น (Transient Command)

1. คำสั่งชนิดที่อยู่ใน CCP (Built-in command) เป็นคำสั่งที่มีโปรแกรมอยู่ภายในส่วนของ CCP เรียบร้อยแล้ว ดังนั้นเมื่อใช้คำสั่งเหล่านี้ก็สามารถทำงานได้เลยทันที คำสั่งที่เป็นคำสั่งชนิดนี้ และที่ใช้ภายใน CP/M มีอยู่ด้วยกัน 5 คำสั่งดังนี้คือ

1.1 DIR เป็นคำสั่งที่ใช้เพื่อให้ทำการแสดงชื่อของไฟล์ข้อมูลต่าง ๆ ที่มีอยู่ภายในดิสค์แผ่นนั้นขึ้นบนจอภาพ โดยโปรแกรมนี้จะทำการนำค่าที่อยู่ภายในไบท์ที่ 2 ถึงไบท์ที่ 12 ของทุก ๆ FCB ที่อยู่ภายในกรุ๊ฟที่ศูนย์และกรุ๊ฟที่หนึ่งที่มีไบท์แรกเป็น 00 (เลขฐานสิบหก)

1.2 ERA คือคำสั่งเพื่อใช้ในการลบชื่อในการลบชื่อของไฟล์ข้อมูลนั้นออกจากแผ่นดิสค์ โดยจะทำการเขียนค่า E5 (เลขฐานสิบหก) ลงไปยังไบท์แรกของ FCB ของแฟ้มข้อมูลนั้น

1.3 REN เป็นคำสั่งที่ใช้ในการเปลี่ยนชื่อของไฟล์ข้อมูลจากชื่อหนึ่งไปเป็นชื่อใหม่อีกชื่อหนึ่งเช่น เมื่อใช้คำสั่งว่า REN KU.COM = YP.COM จะเป็นการเปลี่ยนชื่อไฟล์ข้อมูลที่ชื่อ YP.COM ให้มีชื่อใหม่เป็น KU.COM ดังนี้เป็นต้น

1.4 SAVE คือคำสั่งที่ใช้ในการนำข้อมูลจากหน่วยความจำในส่วนของ TPA ตั้งแต่ TPA base เข้าไปเก็บไว้ในแผ่นดิสค์เช่นเมื่อใช้คำสั่งว่า SAVE 20 KASET.COM จะหมายถึงการนำข้อมูลในหน่วยความจำจากส่วนของ TPA ตั้งแต่แอดเดรส 100 (ฐานสิบหก) เป็นจำนวน 20 เพจ (1 เพจ เท่ากับ 256 ไบท์) เข้าไปเก็บไว้ในไฟล์ข้อมูลที่ให้ชื่อว่า KASET.COM

1.5 TYPE คือคำสั่งที่ใช้ในการนำข้อมูลหรือตัวโปรแกรมในไฟล์ข้อมูลที่มีการเก็บข้อมูลเป็นรหัส ASCII ขึ้นบนจอภาพ เช่น เมื่อใช้คำสั่งว่า TYPE KASET 1.ASM จะเกิดการแสดงตัวโปรแกรมของไฟล์ข้อมูลที่ชื่อ KASET 1 . ASM

2. คำสั่งชนิดที่ผู้ใช้สร้าง (transient command) เป็นคำสั่งที่ผู้ใช้งานสามารถสร้างขึ้นมาได้ในรูปของไฟล์ข้อมูล ๆ หนึ่ง ซึ่งมีชนิดของไฟล์ข้อมูลเป็นแบบ .COM ซึ่งเมื่อทำการเรียกชื่อของไฟล์ข้อมูลดังกล่าวนี้แล้ว เครื่องไมโครคอมพิวเตอร์จะนำคำสั่งนั้นไปทำการตรวจสอบดูว่าเป็นคำสั่งชนิดที่อยู่ภายใน CCP (bulit-in command) หรือไม่ ถ้าใช่ก็จะทำงานตามคำสั่งนั้น แต่ถ้าไม่ใช่ก็จะทำการตรวจสอบชื่อของไฟล์ข้อมูลเป็นชนิด .COM หรือไม่ ถ้ามีก็จะทำการโหลดไฟล์ข้อมูลนั้นลงสู่หน่วยความจำในส่วนของ TPA

แล้วเริ่มต้นทำงานตามแฟ้มข้อมูลนั้นที่แอดเดรส 100 (ฐานสิบหก) แต่ถ้าไม่มีไฟล์ข้อมูลที่มีชื่อตามคำสั่งนั้นจะถือว่าคำสั่งนั้นเป็นคำสั่งที่ผิดพลาด หลังจากนั้นจะเกิดการพิมพ์คำสั่งใหม่อีกครั้งแล้วตามด้วยเครื่องหมาย " ? "

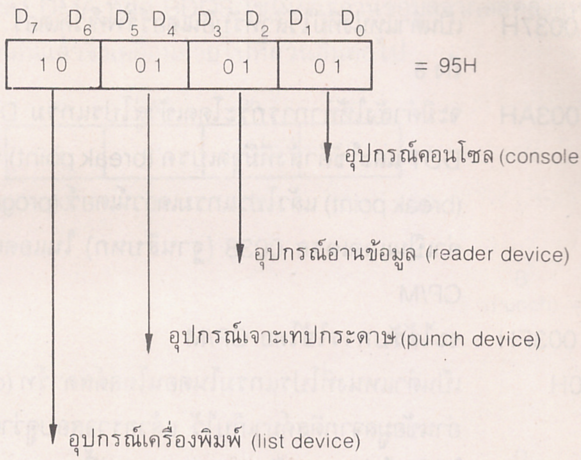
TPA

TPA (the transient program area) เป็นส่วนในหน่วยความจำที่เริ่มตั้งแต่แอดเดรสที่ 100 (ฐานสิบหก) ไปจนถึงแอดเดรสที่ 28FF + b (ฐานสิบหก) ซึ่งจะต่อกับส่วนของจุดเริ่มต้นของ CCP TPA เป็นส่วนที่มีไว้ในการทำงานต่าง ๆ ของโปรแกรม

พื้นที่แสดงพารามิเตอร์ของระบบ (SYSTEM PARAMETER AREA)

พื้นที่แสดงพารามิเตอร์ของระบบเป็นส่วนหนึ่งของหน่วยความจำที่เริ่มตั้งแต่แอดเดรสที่ 00H (ฐานสิบหก) ไปจนถึงแอดเดรสที่ FFH (ฐานสิบหก) หรือเป็นพื้นที่ซึ่งจะถูกใช้ในขณะที่ CP/M กำลังทำงานอยู่ โดยที่แต่ละตำแหน่งจะมีความหมายแตกต่างกันออกไปดังนี้

ตำแหน่งที่	ความหมาย
0000H - 0002H	เป็นตำแหน่งที่มีคำสั่งให้ทำการกระโดด (jump) เข้าสู่ BIOS เพื่อทำการวอร์มสตาร์ทโดยจะเป็นคำสั่ง JMP 3E03H + b ในโปรแกรมที่เขียนขึ้นเมื่อต้องการเข้าสู่ BIOS ก็สามารถทำได้ด้วยการใช้คำสั่ง JMP 0000H
0003H	จะเป็นแอดเดรสที่บรรจุไบท์ข้อมูลที่แสดงอินพุตเอาต์พุตตามมาตรฐานอินเทล (IOBYTE) โดยปกติจะมีค่า 95 (ฐานสิบหก) ใน IOBYTE นี้จะแบ่งออกได้เป็น 4 ส่วนดังนี้ คือ บิทที่ 0 และ 1 หมายถึง อุปกรณ์คอนโซล (console device) บิทที่ 2 และ 3 หมายถึง อุปกรณ์อ่านข้อมูล (reader device) บิทที่ 4 และ 5 หมายถึง อุปกรณ์เจาะเทปกระดาษ (punch device) บิทที่ 6 และ 7 หมายถึง อุปกรณ์เครื่องพิมพ์ (list device) * ในส่วนที่เป็นอุปกรณ์คอนโซล (console device) ถ้ามีค่าเป็น 00 จะหมายถึง คอนโซลอินพุต (console input) ใช้ทาง TTY 01 จะหมายถึง คอนโซลอินพุต (console input) ใช้ทาง CRT 10 จะหมายถึง คอนโซลอินพุต (console input) ใช้ทาง TTY 11 จะหมายถึง คอนโซลอินพุต (console input) ใช้ทาง UC1 (undefine console) * ในส่วนที่เป็นอุปกรณ์อ่านข้อมูล (reader device) ถ้ามีค่าเป็น 00 จะหมายถึง อุปกรณ์อ่านข้อมูล (reader device) ใช้ทาง TTY 01 จะหมายถึง อุปกรณ์อ่านข้อมูล (reader device) ใช้ทาง CRT 10 จะหมายถึง อุปกรณ์อ่านข้อมูล (reader device) ใช้ทาง UR1 (undefine



ตารางที่ 10.3 แสดงลักษณะการเก็บของ IOBYTE

Reader 1)

11 จะหมายถึง อุปกรณ์อ่านข้อมูล (reader device) ใช้ทาง UR2

* ในส่วนที่เป็นอุปกรณ์เจาะเทปกระดาษ (punch device) ถ้ามีค่าเป็น

00 จะหมายถึง อุปกรณ์เจาะเทปกระดาษ (punch device) ใช้ทาง TTY

01 จะหมายถึง อุปกรณ์เจาะเทปกระดาษ (punch device) ใช้ทาง CRT

10 จะหมายถึง อุปกรณ์เจาะเทปกระดาษ (punch device) ใช้ทาง UP1 (undefine punch 1)

11 จะหมายถึง อุปกรณ์เจาะเทปกระดาษ (punch device) ใช้ทาง UP2

* ในส่วนที่เป็นอุปกรณ์เครื่องพิมพ์ (list device) ถ้ามีค่าเป็น

00 จะหมายถึง อุปกรณ์เครื่องพิมพ์ (list device) ใช้ทาง TTY

01 จะหมายถึง อุปกรณ์เครื่องพิมพ์ (list device) ใช้ทาง CRT

10 จะหมายถึง อุปกรณ์เครื่องพิมพ์ (list device) ใช้ทาง TTY

11 จะหมายถึง อุปกรณ์เครื่องพิมพ์ (list device) ใช้ทาง UL1 (undefine list1)

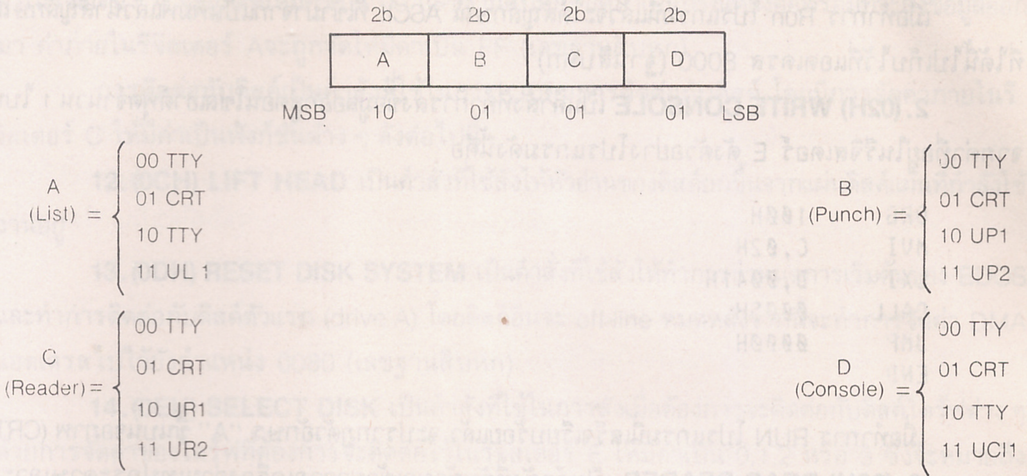
รูปลักษณะของการจัดโครงสร้างและตำแหน่งต่าง ๆ ของ IOBYTE ได้แสดงไว้แล้วในรูปที่ 2.2

0004H — เป็นแอดเดรสที่เก็บค่าที่แสดงถึงดิสค์ไดร์ฟ ที่กำลังติดต่ออยู่กับ BDOS (logged on disk) โดยถ้าเป็นดิสค์ไดร์ฟ A,B,C หรือ D ค่าที่อยู่ภายในแอดเดรสนี้จะมีค่าเป็น 0 1 2 และ 3 ตามลำดับ

0005H - 0007H — เป็นแอดเดรสที่บรรจุคำสั่ง JMP 3106H + b ซึ่งเป็นคำสั่งที่ใช้ในการสั่งให้ทำการกระโดดเข้าสู่ทางเข้า BDOS ซึ่งจะกล่าวถึงต่อไป ที่แอดเดรสนี้จะถูกเปลี่ยนไปเป็น JMP 2100 + b เมื่อโปรแกรม DDT ถูกโหลดเข้าสู่หน่วยความจำและกำลังถูกใช้งานอยู่

0008H - 0037H	เป็นตำแหน่งที่มีไว้สำหรับอินเตอร์รัพท์เวคเตอร์ (interrupt vector) ตำแหน่งที่ 1 ถึง 6
0038H - 003AH	จะมีคำสั่งให้ทำการกระโดดเข้าสู่โปรแกรม DDT เมื่อกำลัง RUN โปรแกรม DDT และใช้คำสั่งที่มีจุดเบรค (break point) ซึ่งเมื่อ RUN ไปจนพบจุดเบรค (break point) แล้วโปรแกรมเคาวน์เตอร์ (program counter) จะได้รับการให้มีค่าเป็นแอดเดรส 0038 (ฐานสิบหก) ในแอดเดรสนี้โดยปกติจะไม่ถูกใช้โดย CP/M
003BH - 003FH	จะได้รับการใช้โดย CP/M
0040H	เป็นตำแหน่งที่โปรแกรมในตอนโคลด์สตาร์ท (cold start) จะใช้เป็นตำแหน่งที่อ่านข้อมูลจากดิสค์มาเก็บไว้ แล้วตรวจสอบดูว่าเป็นดิสค์ของระบบหรือไม่ ถ้าไม่ใช่ก็จะไม่ปรากฏข้อความว่า CP/M ขึ้นบนจอภาพ แต่ถ้าใช่ในไบท์นี้ก็จะถูกจัดให้มีค่าเป็น 30 (ฐานสิบหก)
0041H	เป็นตำแหน่งที่เก็บค่าไบท์บนของ BIOS แอดเดรส ในตำแหน่งนี้เป็นตำแหน่งที่สำคัญที่ใช้ในการโหลด CP/M ลงสู่หน่วยความจำตามตำแหน่งที่ได้ไบแอส (bias) ไว้ โดยค่าภายในไบท์นี้จะเป็นตัวแฟกเตอร์ในการหาค่าไบแอส
0042H	เป็นตำแหน่งที่เก็บไบท์ที่เก็บค่าไดร์ฟของดิสค์ที่จะทำการอ่านหรือเขียนในครั้งต่อไป
0043H	เป็นตำแหน่งที่เก็บไบท์ที่แสดงถึงไดร์ฟของดิสค์ที่ติดต่อกับ BDOS อยู่ก่อนที่จะมีการอ่านหรือเขียนกับดิสค์ในครั้งหลังสุด
0044H	เป็นตำแหน่งที่ถูกใช้ชั่วคราวในกรณีที่เกิดการย้ายดิสค์จากไดร์ฟหนึ่งไปยังอีกไดร์ฟหนึ่ง
0045H - 0046H	เป็นตำแหน่งที่ใช้ในการเก็บค่า DMA แอดเดรสสำหรับการเขียนหรืออ่านจากดิสค์ในครั้งต่อไป
0047H	เป็นตำแหน่งที่เก็บค่าของเซกเตอร์ที่จะทำการเขียนหรืออ่านกับดิสค์ในครั้งต่อไป
0048H	เป็นตำแหน่งที่เก็บค่าของแทร็คที่จะทำการเขียนหรืออ่านกับดิสค์ในครั้งต่อไป
0049H	โดยปกติจะมีค่าเป็น 00 (ฐานสิบหก) และจะเปลี่ยนเป็นค่าอื่นก็ต่อเมื่อเกิดการผิดพลาดขึ้นในการเขียนหรืออ่านจากดิสค์
004AH - 004BH	เป็นตำแหน่งที่ใช้ในการเก็บค่าจุดเริ่มเข้าระบบ โดยปกติจะมีค่าเป็น 00FFH
004CH - 004EH	ไม่ได้ใช้ โดยปกติจะมีค่าเป็น 00H
004FH	เป็นแอดเดรสของ I/O แฟลก โดยปกติจะเก็บค่า 02 (ฐานสิบหก)
0050H - 005BH	จะได้รับการใช้เป็นที่พักข้อมูลโดย CP/M
005CH - 007CH	เป็นตำแหน่งที่ใช้เก็บ FCB (File Control Block)
007DH	โดยปกติจะเก็บคำสั่ง PUSH PSW (F1 เลขฐานสิบหก)
007EH - 007FH	จะเก็บแอดเดรสแรกที่จะถูกใช้งานหลังจากที่ได้ทำงานตามโปรแกรมโหลดเดอร์ในส่วนของ EPROM (warm start) เรียบร้อยแล้ว

0080H - 00FFH เป็นส่วนของ DMA ที่ถูก BDOS ใช้ในการอ่านข้อมูลจากดิสค์ลงสู่หน่วยความจำส่วนนี้ก่อนแล้วจึงเคลื่อนย้ายไปที่ส่วนอื่นต่อไป



รูปที่ 10.4 แสดงความหมายของแต่ละบิท IOBYTE

การเรียกเข้าหาระบบ

ในส่วนของหน่วยความจำในเพจแรกจะมีแอดเดรสอยู่ตำแหน่งหนึ่งที่จะทำหน้าที่เสมือนกับเป็นทางผ่านเข้าสู่ BDOS แอดเดรสที่กล่าวถึงนี้คือแอดเดรสที่ 0005H ถึงแอดเดรสที่ 0007H (เลขฐานสิบหก)

ในการเขียนโปรแกรมด้วยภาษาแอสเซมบลี้ให้ทำการ call ไปยังจุดทางเข้า (entry point) นี้ โดยที่ได้มีการจัดค่าฟังก์ชันต่าง ๆ ในรีจิสเตอร์ C และพารามิเตอร์ข้อมูลหรือแอดเดรสลงในคู่รีจิสเตอร์ DE,B จะสามารถติดต่อกับส่วนของเพอริเฟอรัลต่าง ๆ ได้ โดยไม่ต้องกระโดดเข้าสู่ BIOS

ในการ call นี้เมื่อสิ้นสุดการ call แล้วค่าต่าง ๆ ในรีจิสเตอร์อาจจะมีการเปลี่ยนแปลงได้

ในการ call เข้าสู่จุดทางเข้าของ BDOS นี้ สามารถแบ่งออกได้เป็น 2 พวกคือ

ก. การติดต่อกับอุปกรณ์อินพุตเอาต์พุต

ข. การติดต่อกับดิสค์ (Disk Access primitives)

ก. การติดต่อกับ I/O เป็นคำสั่งที่ใช้ในการติดต่อกับ I/O ต่าง ๆ โดยการจัดค่าของรีจิสเตอร์ตามฟังก์ชันต่าง ๆ ดังต่อไปนี้

1. (01H) READ CONSOLE (with echo) เป็นคำสั่งที่ใช้ในการสั่งให้รับข้อมูลที่เป็นสัญลักษณ์ ASCII จากคอนโซล จำนวน 1 ไบท์เข้าสู่รีจิสเตอร์ แล้วส่งสัญลักษณ์นั้นกลับไปยังคอนโซลอีกครั้งหนึ่ง ตัวอย่างของการใช้คำสั่งนี้มีดังต่อไปนี้ ซึ่งจะเป็นโปรแกรมที่เขียนด้วยภาษา 8080 แอสเซมบลี้

```
ORG     100H
MVI     C,01H
CALL    0005H
LXI     H,8000H
MVI     M,A
```

```
JMP     0000H
END
```

เมื่อทำการ Run โปรแกรมนี้แล้วจะได้สัญลักษณ์ ASCII ที่เข้ามาจากแป้นพิมพ์แล้วนำสัญลักษณ์ที่ได้นี้ไปเก็บไว้ที่แอดเดรส 8000 (ฐานสิบหก)

2. (02H) WRITE CONSOLE เป็นคำสั่งที่ทำการส่งข้อมูลออกสู่คอนโซลเอาต์พุตจำนวน 1 ไบท์จากค่าที่อยู่ในรีจิสเตอร์ E ดังตัวอย่างโปรแกรมดังนี้คือ

```
ORG     100H
MVI     C,02H
LXI     D,0041H
CALL    0005H
JMP     0000H
END
```

เมื่อทำการ RUN โปรแกรมนี้เสร็จเรียบร้อยแล้ว จะปรากฏตัวอักษร ''A'' ขึ้นบนจอภาพ (CRT)

3. (03H) READ READER เป็นคำสั่งที่รับข้อมูลเข้ามาจากเครื่องอ่านเทปกระดาษเจาะรู (paper tape reader) คล้ายกับการอ่านคอนโซล (Read console)

4. (04H) WRITE PUNCH ทำการส่งสัญลักษณ์ที่เป็นค่าที่อยู่ภายในรีจิสเตอร์ E ออกทางเอาต์พุตเทปกระดาษเจาะรู คล้ายกับการเขียนคอนโซล (Write console)

5. (05H) WRITE LIST เป็นการส่งข้อมูลที่อยู่ภายในรีจิสเตอร์ E ออกทางเครื่องพิมพ์ 1 ตัวอักษร (1 ไบท์)

7. (07H) GET I/O STATUS เป็นการเคลื่อนย้ายเอาค่า IOBYTE จากแอดเดรส 0003H มาไว้ที่รีจิสเตอร์ A

8. (08H) SET I/O STATUS เป็นการจัดค่าภายใน IOBYTE (แอดเดรสที่ 0003 ฐานสิบหก) ใหม่ให้มีค่าเป็นไปตามค่าที่อยู่ภายในรีจิสเตอร์ Eดังตัวอย่างเช่น

```
ORG     100H
MVI     C,08H
LXI     D,0094H
CALL    0005H
END
```

เมื่อทำการ RUN โปรแกรมนี้เรียบร้อยแล้ว ค่าใน IOBYTE จะถูกเปลี่ยนไปเป็น 94 (ฐานสิบหก) ดังนี้เป็นต้น

9. (09H) PRINT BUFFER เป็นการสั่งให้ทำการส่งข้อมูลที่อยู่ในหน่วยความจำตำแหน่งที่มีแอดเดรสที่ตรงกับข้อมูลที่อยู่ภายในรีจิสเตอร์ DE ไปจนกว่าจะพบสัญลักษณ์ของตัวอักษร $

10. (0AH) READ BUFFER เป็นการอ่านข้อมูลที่เป็นสตริงจากคอนโซลอินพุต จำนวนหนึ่งเข้าไปทำการเก็บไว้ภายในหน่วยความจำยังตำแหน่งแอดเดรสที่ตรงกับค่าที่อยู่ภายในคู่รีจิสเตอร์ DE บวก 2 โดยที่แอดเดรสที่ตรงกับค่าในคู่รีจิสเตอร์ DE จะเป็นตัวบอกจำนวนไบท์ที่มากที่สุดของสตริงนี้ และค่าไบท์ต่อมาจะบอกให้รู้ถึงจำนวนไบท์ที่ถูกใช้ไป หลังจากสิ้นสุดการเติมค่านี้

11. (0BH) INTERROGATE CONSOLE READY เป็นการตรวจสอบสถานะของคอนโซลว่าอยู่ในสภาพหรือสถานะที่พร้อมที่จะรับข้อมูลเข้าไปหรือพร้อมที่จะส่งข้อมูลออกมา โดยถ้าเป็นกรณีแรกค่าในรีจิสเตอร์ A จะถูกจัดให้มีค่าเป็น 00 (เลขฐานสิบหก) แต่ถ้าเป็นกรณีหลังคือพร้อมที่จะส่งข้อมูลออกมา ค่าภายในรีจิสเตอร์ Aจะถูกจัดให้มีค่าเป็น FF (เลขฐานสิบหก)

การติดต่อกับดิสค์เป็นคำสั่งที่ใช้ในการควบคุมการติดต่อกับดิสค์ โดยมีการจัดค่าภายในรีจิสเตอร์ C ให้มีค่าเป็นฟังก์ชันต่าง ๆ ดังต่อไปนี้

12. (0CH) LIFT HEAD เป็นคำสั่งที่ใช้สั่งให้หัวอ่านของดิสค์ยกขึ้นจากแผ่นดิสค์แผ่นที่กำลังใช้งานอยู่

13. (0DH) RESET DISK SYSTEM เป็นคำสั่งที่ใช้สั่งให้ทำการกำหนดการเริ่มต้นของ BDOS และทำการติดต่อกับดิสค์ตัวแรก (drive A) โดยดิสค์อื่นจะ off-line หมดหลังจากนี้จะทำการจัดค่า DMA แอดเดรสไปไว้ยังตำแหน่ง 0080 (เลขฐานสิบหก)

14. (0EH) SELECT DISK เป็นคำสั่งที่ใช้ในการสั่งเมื่อต้องการจะติดต่อกับดิสค์ไดร์ฟต่าง ๆ ด้วยการจัดค่าของไดร์ฟที่ต้องการจะติดต่อไว้ในรีจิสเตอร์ E ให้มีค่าเป็น 0,1,2 หรือ 3 ซึ่งจะหมายถึง drive A, B, C หรือ D ตามลำดับ

15. (0FH) OPEN FILE ทำการติดต่อกับ FCB (File control block) ของไฟล์ข้อมูลที่มี FCB ตรงกับ FCB ที่จำลองขึ้นมาอยู่ภายในหน่วยความจำที่แอดเดรสที่ตรงกับค่าที่อยู่ภายในคู่รีจิสเตอร์ DE

ถ้า BDOS ไม่สามารถจะหาไฟล์ข้อมูลนั้นได้ก็จะทำการจัดค่าในรีจิสเตอร์ A ให้มีค่าเป็น FF (เลขฐานสิบหก) แต่ถ้าในการติดต่อนี้สามารถติดต่อได้ก็จะทำการจัดค่าในรีจิสเตอร์ A ให้มีค่าเป็น 00 (เลขฐานสิบหก)

16. (10H) CLOSE FILE คู่รีจิสเตอร์ DE จะถูกจัดให้มีค่าเป็นแอดเดรสของ FCB ของไฟล์ข้อมูลที่ต้องการจะทำการปิด เมื่อทำคำสั่งนี้แล้ว BDOS จะทำการเปลี่ยนแปรค่าในส่วนของ DM ใน FCB ใหม่ด้วย

ถ้า BDOS ไม่สามารถจะหา FCB ของไฟล์ข้อมูลนั้นได้จะทำการจัดค่าภายในรีจิสเตอร์ A ให้มีค่าเป็น FF (เลขฐานสิบหก) และถ้าสามารถติดต่อได้ก็จะทำการจัดค่าภายในรีจิสเตอร์ A ให้มีค่าเป็น 00 (เลขฐานสิบหก)

17. (11H) SEARCH FILE เป็นคำสั่งที่ใช้ในการค้นหา FCB ของไฟล์ข้อมูลที่ต้องการ ที่ทำการเขียนไว้ในแอดเดรสที่ตรงกับค่าที่อยู่ภายในคู่รีจิสเตอร์ DE ในคำสั่งนี้จะมีลักษณะขั้นตอนการทำงานเหมือนกับการเปิดแฟ้มข้อมูลหรือไฟล์

ในการค้นหาไฟล์ข้อมูลโดยใช้คำสั่งนี้จะทำการค้นหาเฉพาะไฟล์ข้อมูลที่มีค่าในไบท์แรกของ FCB ไม่เป็นค่า E5 (เลขฐานสิบหก) เท่านั้น ทั้งนี้เพราะว่าไฟล์ข้อมูลที่ถูกลบทิ้งไปแล้วจะถูกเขียนค่า E5 (เลขฐานสิบหก) ลงไปยังไบท์แรกของ FCB ของไฟล์ข้อมูลนั้น คำสั่งในการค้นหาไฟล์ข้อมูลนี้จะทำการค้นหา FCB แรก (exten แรก) เท่านั้น หลังจากสิ้นสุดคำสั่งนี้แล้ว จะต้องตามด้วยคำสั่งยกหัว (12) เสมอ

18. (12H) SEARCH NEXT FILE เป็นคำสั่งที่เหมือนกับการค้นหาไฟล์ข้อมูลในคำสั่งที่ 17 แต่จะทำการค้นหา FCB ต่อไป (exten ต่อไป ของไฟล์ข้อมูลนั้น ๆ)

19. (13H) DELETE FILE เป็นคำสั่งที่ใช้ในการลบชื่อของไฟล์ข้อมูลใดไฟล์ข้อมูลหนึ่งออกจากสารบัญโดยค่าที่อยู่ในคู่รีจิสเตอร์ DE จะเป็นแอดเดรสภายในหน่วยความจำที่เก็บค่าสัญลักษณ์ ASCII ของชื่อไฟล์ข้อมูลและชนิดข้อมูลที่ต้องการจะทำการลบ

20. (14H) READ NEXT RECORD เป็นคำสั่งที่ใช้ในการอ่านข้อมูลจากแผ่นดิสค์โดยค่าในคู่รีจิสเตอร์ DE จะเป็นค่าแอดเดรสของ FCB ที่ได้ถูกเปิดไว้เรียบร้อยแล้ว เรคอร์ดที่จะทำการอ่านทั้ง 128 ไบท์จะถูกอ่านเข้าสู่หน่วยความจำตำแหน่งที่ได้ทำการจัดค่า DMA แอดเดรสไว้แล้ว ซึ่งโดยปกติแล้วจะเป็นที่แอดเดรสที่ 0080 ถึงแอดเดรส 00FF (เลขฐานสิบหก) และในไบท์ NR ของ FCB จะถูกเพิ่มค่าขึ้นอีก 1 เพื่อทำการอ่านเซกเตอร์ต่อไปหลังจากสิ้นสุดคำสั่งนี้แล้วค่าในรีจิสเตอร์ A จะถูกจัดให้มีค่าเป็นค่าใดค่าหนึ่งใน 3 ค่าต่อไปนี้คือ

0 ศูนย์ จะหมายถึง การอ่านได้เรียบร้อย

1 จะหมายถึง การอ่านได้จบไฟล์ข้อมูลนั้นแล้ว

2 จะหมายถึง ว่าได้เกิดการผิดพลาดในการอ่านในครั้งนั้น

21. (15H) WRITE NEXT RECORD เป็นคำสั่งที่ใช้ในการสั่งให้ทำการเขียนข้อมูลจากหน่วยความจำเป็นจำนวน 128 ไบท์ที่แอดเดรสที่ได้ทำการจัดค่า DMA แอดเดรสไว้แล้วให้เข้าสู่ดิสค์ยังไฟล์ข้อมูลที่มี FCB ที่ได้ทำการเปิดไว้แล้ว และมีคู่รีจิสเตอร์ DE เก็บแอดเดรสของ FCB นี้ในหน่วยความจำใน NR ไบท์ของ FCBจะถูกเปลี่ยนไปเป็นตัวเลขของเรคอร์ด

หลังจากที่สิ้นสุดคำสั่งนี้แล้วค่าในรีจิสเตอร์ A จะถูกจัดให้เป็นค่าใดค่าหนึ่งใน 4 ค่าต่อไปนี้คือ

0 หมายถึง การเขียนในครั้งนั้นประสบผลสำเร็จ

1 หมายถึง ได้เกิดการผิดพลาดบางอย่างในการเขียนนี้

2 หมายถึง ว่าไฟล์ข้อมูลนั้นได้สิ้นสุดแล้ว (end of disk data)

-1 หมายถึง ไม่มีที่ว่างเหลือสำหรับสารบัญ

22. (16H) MARK FILE เป็นคำสั่งที่ใช้ในการสร้างไฟล์ข้อมูลขึ้นมาโดยมีชื่อของไฟล์ข้อมูลและชนิดของไฟล์ข้อมูลที่อยู่ภายใน FCB ที่อยู่ที่แอดเดรสที่ตรงกับค่าที่อยู่ภายในคู่รีจิสเตอร์ DE และในส่วนของ DM ใน FCB นี้จะถูกจัดให้มีค่าเป็น 00 (เลขฐานสิบหก)

หลังจากสิ้นสุดคำสั่งนี้แล้ว ค่าภายในรีจิสเตอร์ A จะถูกจัดให้มีค่าเป็นค่าของจำนวนจุดเข้าของสารบัญ (directory entry) และจะมีค่าเป็นลบหนึ่งถ้าไม่มีที่ว่างเหลือภายในส่วนของสารบัญ

23. (17H) RENAME FCB เป็นคำสั่งที่ทำการเปลี่ยนทั้งชื่อของไฟล์ข้อมูลและชนิดของไฟล์ข้อมูลทั้ง 16 ไบท์แรกของ FCB ที่ถูกอ้างถึงโดยคู่รีจิสเตอร์ DE ไปเป็นชื่อไฟล์ข้อมูลใหม่ตามค่าภายใน 16 ไบท์หลังของ FCB นั้น

ถ้าการเปลี่ยนชื่อในครั้งนั้นประสบผลสำเร็จ ค่าภายในรีจิสเตอร์ A จะถูกจัดให้มีค่าเป็นจำนวนจุดเข้าของไฟล์ข้อมูลที่ถูกเปลี่ยนชื่อไปและถ้าในกรณีที่มีการผิดพลาดเกิดขึ้น ค่าภายในรีจิสเตอร์ A จะถูกจัดให้มีค่าเป็น -1 (255 หรือ 0FF เลขฐานสิบหก)

24. (18H) INTERROGATE LOGIN VECTOR เป็นคำสั่งที่ใช้เมื่อต้องการจะทราบถึงไดร์ฟที่ BDOS กำลังทำการติดต่ออยู่ โดยเมื่อสิ้นสุดคำสั่งนี้แล้ว ค่าภายในรีจิสเตอร์ A จะเป็นตัวบอกถึงค่าของไดร์ฟ

ของดิสค์ตัวที่กำลังทำการติดต่ออยู่ โดยถ้าบิทที่ 0 ถูกจัดให้มีค่าเป็น 1 ก็จะแสดงถึงว่า BDOS กำลังทำการติดต่ออยู่กับไดร์ฟ A และถ้าบิทที่ 1 ถูกจัดให้มีค่าเป็น 1 ก็จะหมายถึงไดร์ฟ B และถ้าบิทที่ 2 และ 3 ถูกจัดให้มีค่าเป็น 1 ก็จะหมายถึง ไดร์ฟ C และ D ตามลำดับโดยบิทอื่นจะถูกจัดให้มีค่าเป็น 0 หมด

25. (19H) INTERROGATE DRIVE NUMBER เป็นคำสั่งที่ใช้เมื่อต้องการจะทราบถึงจำนวนไดร์ฟที่ BDOS สามารถติดต่อได้ โดยคำตอบจะอยู่ภายในรีจิสเตอร์ A หลังจากสิ้นสุดคำสั่งนี้แล้ว

26. (1AH) SET DMA ADDRESS เป็นคำสั่งที่ใช้ในการจัดค่า DMA แอดเดรสไว้ ซึ่งโดยปกติจะถูกจัดให้มีค่าเป็น 0080 (เลขฐานสิบหก) ค่า DMA แอดเดรสนี้จะถูกจัดให้มีค่าตามค่าที่อยู่ภายในคู่รีจิสเตอร์ DE

27. (1BH) INTERROGATE ALLOCATION เป็นคำสั่งที่ใช้ในการตรวจสถานะต่าง ๆ ของดิสค์ซึ่งจะใช้โดยโปรแกรม STAT.COM เท่านั้น

หลังจากกระทำคำสั่งนี้แล้ว ค่าในคู่รีจิสเตอร์ BC จะแสดงถึงแอดเดรสของตารางบิท (bit map) ของกรุ๊ฟของดิสค์นั้น และค่าภายในคู่รีจิสเตอร์ BC จะแสดงถึงจำนวนกรุ๊ฟที่ถูกใช้ไปในดิสค์แผ่นนั้น และค่าในรีจิสเตอร์ A จะแสดงถึงจำนวนเซกเตอร์ต่อ 1 กรุ๊ฟที่ใช้ ซึ่งโดยปกติจะเท่ากับ 08 (เลขฐานสิบหก)

โครงสร้างการเชื่อมต่อโดยผ่านพารามิเตอร์ตามคำสั่งที่กล่าวมาแล้วทั้งหมดเป็นวิถีทางที่จะทำให้ผู้ใช้สามารถติดต่อกับ CP/M ได้ง่าย โปรแกรมสำเร็จรูปที่มีผู้พัฒนาขึ้นส่วนใหญ่จะเชื่อมต่อกับ CP/M ผ่านทางเหล่านี้

ภาคผนวก ก.

ซอฟต์แวร์ของ CP/M

ชนิดของซอฟต์แวร์

ซอฟต์แวร์แบ่งเป็น 2 ชนิดใหญ่ ๆ คือ *โปรแกรมระบบ (Systems Software)* และ *โปรแกรมประยุกต์ (Applications software)* โปรแกรมระบบนั้นใช้ในการควบคุมการทำงานของระบบคอมพิวเตอร์ เช่น โปรแกรมจัดระบบงาน (Operating System) โปรแกรมโหลดเดอร์ (Loader program) โปรแกรมยูทิลิตี้ (Utility program) และ โปรแกรมภาษาต่าง ๆ ส่วนโปรแกรมประยุกต์นั้นได้รับการพัฒนาขึ้นมาเพื่อที่นำคอมพิวเตอร์ไปใช้งานในการแก้ปัญหาต่าง ๆ เช่นการนำคอมพิวเตอร์มาใช้งานทางด้านการประมวลผลข้อมูลเป็นต้น

ในบทนี้จะกล่าวถึงส่วนของโปรแกรมระบบที่สามารถทำงานภายใต้ CP/M ได้

โปรแกรมยูทิลิตี้ (Utility Programs)

เป็นโปรแกรมต่าง ๆ ที่ช่วยอำนวยความสะดวกในการใช้เครื่องคอมพิวเตอร์ที่สำคัญ และใช้เสมอ ๆ ได้แก่ เอดิเตอร์ (ED) และไดนามิก ดีบักกิ้ง ทูล (Dynamic Debugging Tool) (DDT) เป็นต้น ซึ่งทั้งสองโปรแกรมนี้มีลักษณะการใช้งานดังนี้

1. เอดิเตอร์ (ED) เป็นโปรแกรมที่ใช้สำหรับสร้างหรือแก้ไขโปรแกรมที่เขียนขึ้นเพื่อที่จะสั่งให้เครื่องคอมพิวเตอร์ทำงานโดยผ่านทางคีย์บอร์ด และจอภาพ (CRT) เมื่อทำการสร้างหรือแก้ไขโปรแกรมเรียบร้อยแล้ว ก็จะนำโปรแกรมนี้เข้าไปเก็บเป็นแฟ้มข้อมูลในแผ่นจานแม่เหล็ก

ในการเรียกใช้โปรแกรมเอดิเตอร์ (ED) นี้ ใช้เรียกพร้อมทั้งการตั้งชื่อแฟ้มข้อมูลหรือชื่อของตัวโปรแกรมที่จะทำการเขียนในกรณีที่เริ่มทำการเขียนโปรแกรมนั้นเป็นครั้งแรก หรือเป็นการสร้างโปรแกรมขึ้นมานั่นเอง แต่ถ้าเป็นการแก้ไขโปรแกรมที่มีอยู่แล้ว ก็จะเป็นการนำโปรแกรมนั้นมาโดยทำการโหลดจากแผ่นดิสเก็ตต์ลงสู่หน่วยความจำในเครื่องคอมพิวเตอร์ ตัวอย่างเช่นในการที่จะสร้างโปรแกรมด้วยภาษา 8080 แอสเซมบลี้ ให้ทำการพิมพ์ ตัวอักษรที่จะเป็นชื่อของโปรแกรมนั้นทางคีย์บอร์ด โดยให้ชื่อซอร์สโปรแกรมว่า KU.ASM จะมีขั้นตอนในการเขียนดังนี้

ก. เริ่มต้นตั้งชื่อของซอร์สโปรแกรมและเข้าสู่โหมดของ ED ดังนี้

ED KU.ASM cr

หลังจากนี้เครื่องจะพิมพ์คำว่า "NEW FILE" และจะพิมพ์เครื่องหมาย * ซึ่งจะหมายถึงการพร้อมที่จะรับคำสั่งต่อไป

ข. สั่งให้เครื่องพร้อมที่จะรับโปรแกรมด้วยคำสั่ง "VI" ซึ่งจะหมายถึงการ verify คือแสดงตัวเลขบรรทัดไว้ด้วย และคำสั่ง Insert ซึ่งเป็นการเริ่มต้นการเขียนโปรแกรม

ค. ทำการเขียนโปรแกรมลงไปดังนี้

```
        ORG     100H    ; THE ORGIN OF THIS PROGRAM
START   LXI     D,0041H ; MOV E-REGISTER WITH ASCII 'A'
        MVI     C,05H   ; SET SYSTEM CALL FUNCTION
        CALL    0005H
        END     PROGRAM
```

ง. การสิ้นสุดการเขียนโดยการกดปุ่มคอนโทรล และปุ่ม Z พร้อมกัน จะปรากฏเครื่องหมาย " * " บนจอภาพ

จ. ทำการตรวจทานโปรแกรมโดยใช้คำสั่ง "B5T" ซึ่งเป็นการแสดงตัวซอร์สโปรแกรมตั้งแต่บรรทัดแรกต่อไป 5 บรรทัด

ฉ. ทำการเก็บซอร์สโปรแกรมลงไปยังแผ่นดิสค์โดยใช้คำสั่ง "E" ซึ่งจะเป็นการออกจากโปรแกรมเอดิเตอร์ (ED) ด้วย

ในการเรียกโปรแกรม KU.ASM มาแก้ไขก็ทำการเรียกในลักษณะเดิมอีกเช่นกัน ซึ่งรายละเอียดดูได้จากคู่มือเอดิเตอร์ (ED)

ในการตั้งชื่อของซอร์สโปรแกรมต้องตั้งชื่อให้มีความยาวไม่เกิน 8 ตัวอักษร ส่วนชนิดของโปรแกรมจะมีได้ 3 ตัวอักษรเท่านั้น และที่ใช้มีดังนี้

ASM	เป็น 8080 แอสเซมบลี้โปรแกรม
BAK	เป็นโปรแกรมสำรอง (Bakup)
BAS	เป็นโปรแกรมภาษาเบสิค
CMD	เป็นโปรแกรมที่เขียนเป็นคำสั่งในการทำงานของเครื่องใช้คู่กับ d BASE
COB	เป็นโปรแกรมภาษาโคบอล
COM	เป็นโปรแกรมที่ถูกแปลงเป็นภาษาเครื่องแล้ว (machine code)
FOR	เป็นโปรแกรมภาษาฟอร์แทรน 4
HEX	เป็นโปรแกรมที่ถูกแปลงเป็นเลขฐาน 16
INT	เป็นอินเตอร์มีเดียต (intermediate) โปรแกรมของภาษาเบสิค
PRN	เป็นโปรแกรมภาษาแอสเซมบลี้ ที่มีรหัสเลขฐาน 16 กำกับอยู่
REL	เป็นรีโลเคเตเบิ้ล (Relocatable) โปรแกรม
SUB	เป็นโปรแกรมที่เขียนเป็นคำสั่งในการทำงานของเครื่อง จะใช้คู่กับโปรแกรม SUMBIT เช่นเดียวกับ BAT
Z80	เป็นโปรแกรมที่เขียนด้วยภาษา Z80 แอสเซมบลี้

2. โปรแกรมที่ใช้ในการแก้ไขโปรแกรม (debugging program) เป็นโปรแกรมที่ช่วยในการทดสอบโปรแกรมภาษาแอสเซมบลี้ และช่วยหาข้อผิดพลาดของโปรแกรม ในระบบ CP/M มีโปรแกรมในการแก้ไขนี้มีชื่อว่าไดนามิก ดีบักกิ้ง ทูล (DDT) ซึ่งมีคำสั่งในการใช้งานอยู่หลายคำสั่งเช่น

A เพื่อทำการเขียนค่ารหัสนีโมนิก (mnemonic code) ของภาษา 8080 แอสเซมบลี้ลงไปยังแอดเดรสที่ต้องการ

D ทำการแสดงค่าในหน่วยความจำออกทางจอภาพเป็นตัวเลขฐานสิบหกและเป็นรหัส ASCII

L ทำการแสดงค่าในหน่วยความจำออกทางจอภาพเป็นสัญลักษณ์นีโมนิกของภาษา 8080 แอสเซมบลี้

S เป็นการแก้ไขค่าที่อยู่ในหน่วยความจำ

T ใช้ในการเทรซ ดูผลของการที่ซีพียูทำงานแต่ละคำสั่ง

G เป็นคำสั่งที่ใช้เพื่อให้คอมพิวเตอร์เริ่มทำงานตามแอดเดรสที่ต้องการ

รายละเอียดของคำสั่งต่าง ๆ ภายใน DDT ดูได้จากคู่มือ DDT

โปรแกรมประมวลผลภาษา (Language Processor)

เป็นโปรแกรมที่ใช้ในการแปลข้อมูลที่อยู่ในรูปแบบของภาษาหนึ่งของคอมพิวเตอร์ไปสู่อีกรูปแบบหนึ่ง โดยอินพุตที่เข้าสู่ตัวโปรแกรมประมวลผลภาษีนี้จะเรียกว่าโปรแกรม ส่วนเอาต์พุตจะเรียกว่าออบเจ็กต์โปรแกรม (Object Program) ตัวโปรแกรมประมวลผลภาษาที่ใช้อยู่กันทั่วไปจะแบ่งออกได้เป็น 3 ระดับคือ แอสเซมบลี้ มาโครแอสเซมบลี้ และภาษาระดับสูง

แอสเซมเบลอร์และมาโครแอสเซมเบลอร์ที่ใช้ใน CP/M จะเป็นโปรแกรมที่เขียนขึ้นด้วยภาษาของ 8080 แอสเซมบลี้ แล้วอาศัยคอมพลายเลอร์ (compiler) ที่ชื่อ ASM.COM ให้ทำการเปลี่ยนจากสัญลักษณ์นีโมนิก ไปเป็นรหัสเลขฐานสองหรือภาษาเครื่อง และทำการจัดรูปของภาษาเครื่องใหม่ให้เป็นโปรแกรมชนิดไฟล์ที่เป็นชนิด COM เพื่อใช้เป็นทรานเชียนต์ คอมมานด์

ในการคอมพลายโปรแกรมที่เขียนขึ้นด้วยภาษาของ 8080 แอสเซมบลี้ จะต้องทำการคอมพลายยังแผ่นที่มีโปรแกรมเหล่านี้อยู่ภายในแผ่นที่สามารถใช้งานได้ คือ

ASM.COM - คอมพลายเลอร์

LOAD.COM - โปรแกรมที่ใช้เปลี่ยนเป็นไฟล์คำสั่ง

โดยมีลักษณะการใช้ดังตัวอย่างต่อไปนี้ซึ่งจะใช้คอมพลายโปรแกรมที่ชื่อว่า KU.ASM ที่อยู่ภายในแผ่นดิสค์ที่มีทั้งสองโปรแกรมอยู่ด้วย ดังนี้คือ

ASM KU.mno - ทำการคอมพลายโปรแกรมชื่อ KU

LOAD KU - ทำการสร้างไฟล์ KU เป็น KU.COM

KU - ทำการ RUN โปรแกรมที่มีชื่อว่า KU

ตัวอักษร m,n และ o เป็นตัวอักษรเดี่ยว ๆ ซึ่งมีความหมายดังนี้

ถ้า m เป็น A,B, C,D แสดงถึงไดร์ฟของดิสค์ที่บรรจุซอร์ตโปรแกรมไว้

ถ้า n เป็น A,B,C,D แสดงถึงไดร์ฟของดิสค์ที่จะบรรจุ HEX ไฟล์
หรือ ถ้า n เป็น Z แสดงว่าไม่ต้องมีการสร้าง HEX ไฟล์
ถ้า o เป็น A,B,C,D แสดงถึงไดร์ฟของดิสค์ที่จะบรรจุ PRN ไฟล์
หรือ ถ้า o เป็น X แสดงว่าให้แสดงไฟล์สำหรับพิมพ์ออกทางจอภาพด้วย
หรือ ถ้า o เป็น Z แสดงว่าไม่ต้องมีการสร้าง PRN ไฟล์

เมื่อคอมพลายโดยไม่ใช้ m,n และ o จะถือว่าทำการคอมพลายโดยเอาซอร์สโปรแกรมจากแผ่นดิสเก็ตต์แผ่นที่กำลังทำการติดต่ออยู่ และสร้างโปรแกรม HEX และ PRN ยังดิสค์แผ่นเดิมนั้น โดยจะแสดงบรรทัดที่มีคำผิดเท่านั้นบนจอภาพ และกำกับด้วยสัญลักษณ์ต่าง ๆ ข้างหน้าบรรทัด เช่น U, V หรือ S เป็นต้น ซึ่งความหมายของสัญลักษณ์ต่าง ๆ เหล่านี้จะดูได้จากคู่มือ ASM

ในภาษาระดับสูงที่สามารถใช้กับ CP/M ได้มีอยู่ด้วยกันหลายภาษาคือ *ภาษาเบสิค ภาษาฟอร์แทน 4 ภาษาโคบอล ฯลฯ*

1. **ภาษาเบสิค** จะแบ่งออกได้เป็น 2 แบบคือ

- *อินเตอร์พรีทเตอร์ (interpreter)*
- *คอมพลายเลอร์ (compiler)*

โดยคอมพลายเลอร์จะทำการแปลโปรแกรมทั้งหมดให้เป็นออบเจ็กต์โปรแกรม ซึ่งประกอบด้วยภาษาเครื่องทั้งหมด และจะเก็บออบเจ็กต์โปรแกรมไว้ที่แห่งหนึ่ง เช่นเป็นไฟล์ข้อมูลภายในแผ่นดิสค์เป็นต้น เมื่อเครื่องคอมพิวเตอร์จะทำการเอ็กซีคิวโปรแกรม ก็จะโหลดเอาออบเจ็กต์โปรแกรมลงไปไว้ในหน่วยความจำแล้วก็ทำงานตามคำสั่งต่าง ๆ ซึ่งเป็นภาษาเครื่องนั้นโดยไม่ต้องยุ่งเกี่ยวกับโปรแกรม ส่วนอินเตอร์พรีทเตอร์นั้นไม่ได้ทำการแปลโปรแกรมทั้งหมดให้เป็นออบเจ็กต์โปรแกรม แต่จะนำทีละคำสั่งที่เขียนไว้ในโปรแกรมมาทำการแปลและเอ็กซีคิวทันทีทีละคำสั่ง ฉะนั้นถ้าคำสั่งใดจะถูกใช้งานหลายครั้งเครื่องคอมพิวเตอร์ก็จะต้องเสียเวลาแปลคำสั่งนั้นทุก ๆ ครั้ง ซึ่งจะเป็นการเสียเวลามากกว่าแบบแรก

อินเตอร์พรีทเตอร์เบสิคนี้ที่สามารถ RUN ภายใต้ CP/M ต้องอาศัยโปรแกรมที่ชื่อ MBASIC.COM ในการที่จะเขียนโปรแกรมด้วยภาษาเบสิคแบบของอินเตอร์พรีทเตอร์ โดยสามารถทำได้ตามตัวอย่างดังต่อไปนี้

MBASIC	- เรียกโปรแกรม MBASIC ลงสู่หน่วยความจำ
LOAD "KU.BAS"	- ทำการเรียกโปรแกรม KU.BAS จากแผ่นดิสค์ลงสู่หน่วยความจำในกรณีที่ต้องการแก้ไขโปรแกรมใหม่
AUTO 200	- สั่งให้แสดงเลขบรรทัดโดยเริ่มตั้งแต่บรรทัดที่ 200 เป็นต้นไป
ctl-C	กดปุ่ม control และ C พร้อมกันจะเป็นคำสั่งที่สั่งให้ออกจากโหมด AUTO
SAVE "KU"	- เป็นการสั่งให้ทำการย้ายโปรแกรมจากหน่วยความจำไปเก็บเป็นไฟล์ข้อมูลที่ชื่อ KU.BAS และไฟล์ข้อมูลนี้จะถูกเก็บไว้ยังแผ่นดิสค์
SYSTEM	- เป็นคำสั่งที่ใช้ในการสั่งให้ออกจากโปรแกรม MBASIC เข้าสู่ CP/M

ส่วนเบสิคที่เป็นคอมพลายจะต้องมีโปรแกรมนี้อยู่คือ

- CBAS2.COM - โปรแกรมคอมพลายเลอร์ให้เป็นรหัสอินเตอร์มีเดียต

- CRUN2.COM - เป็นโปรแกรมทำการ RUN รหัสอินเตอร์มีเดียต

ในเบสิคแบบนี้จะเป็นการผสมระหว่างคอมพลายและอินเตอร์พรีทเตอร์ คือในช่วงแรก เครื่องจะทำการแปลซอร์สโปรแกรมให้เป็นโปรแกรมรหัสอินเตอร์มีเดียตคืออยู่ในรูปของรหัสที่สร้างขึ้นมาใหม่ เพื่อที่จะให้ CRUN2.COM ทำการแปลและเอ็กซีคิวต่อไป ซึ่งเป็นการลดขนาดของโปรแกรมลง ตัวอย่างของการใช้งานจะมีดังนี้

CBAS2 KU - เป็นการคอมพลายโปรแกรมชื่อ KU.BAS

CRUN2 KU - เป็นการอินเตอร์พรีทเตอร์โปรแกรมที่ชื่อว่า KU.INT ที่ถูกสร้างขึ้นจากคำสั่งแรก

2. ภาษาฟอร์แทรน 4 ภาษานี้เป็นภาษาระดับสูงที่ใช้กันอย่างกว้างขวางและใกล้เคียงกับภาษาที่ใช้พูดกัน ดังนั้นในการสร้างคอมพลายเลอร์ของภาษาฟอร์แทรน 4 นี้จึงมีขนาดใหญ่และมีหลายโปรแกรม ซึ่งคอมพลายภาษาฟอร์แทรน 4 ที่ใช้อยู่กับเครื่องไมโครคอมพิวเตอร์ของภาควิชาวิศวกรรมไฟฟ้า มหาวิทยาลัยเกษตรศาสตร์เป็นคอมพลายที่เขียนขึ้นโดยบริษัท TECHNICAL DESIGN LABS,INC. (TDL) และสามารถ RUN ภายใต้ CP/M ได้ แต่ต้องมีโปรแกรมเหล่านี้อยู่ด้วยคือ

FOR.COM - เป็นคอมพลายเลอร์ของภาษาฟอร์แทรน 4

LOADER.COM - เป็นโปรแกรมที่ทำการเชื่อมต่อกับโปรแกรม LIBRARY.REL

LOD.COM - เป็นโปรแกรมที่ใช้สร้าง HEX file.

MLIB.REL - เป็น Library ทางคณิตศาสตร์และทางตรรกวิทยา

LIBRARYS.REL - เป็น System subroutine Library

ในการคอมพลายจะมีตัวอย่างดังนี้ โดยสมมติให้โปรแกรมที่เขียนด้วยภาษาฟอร์แทรน 4 มีชื่อว่า KU.FOR

FOR KU - ทำการคอมพลายโปรแกรม KU.FOR แล้วพิมพ์ตัวโปรแกรมพร้อมทั้งที่ผิดออกทางเครื่องพิมพ์

LOADER KU/L, MLIB - ทำการเชื่อมต่อระหว่างตัวโปรแกรม KU กับ Library และสร้างโปรแกรมที่มีชื่อว่า KU.COM

KU ทำการ RUN โปรแกรม KU

ในการคอมพลายนี้ถ้าเกิดมีข้อผิดพลาดเกิดขึ้น จะปรากฏสัญลักษณ์ของข้อผิดพลาดเป็นตัวเลข (เช่น 002 เป็นต้น) ความหมายของตัวเลขเหล่านี้จะสามารถค้นหาได้จากคู่มือฟอร์แทรน 4 (เล่มที่ 14)

I/O ยูนิท ที่ใช้ในภาษาฟอร์แทรน 4 มีหลายหน่วย แต่ที่ใช้มากมีอยู่ 2 หน่วยคือ

1 หมายถึง CRT (เป็นได้ทั้งอินพุตและเอาต์พุต)

2 หมายถึง เครื่องพิมพ์ (เอาต์พุตได้เพียงอย่างเดียว)

3. ภาษาโคบอล การคอมพลายภาษาโคบอลนี้จะต้องทำภายใต้โปรแกรมจัดระบบงาน TSA OS หรือ CP/M เวอร์ชั่น 2.2 เท่านั้น ซึ่งในแผ่นดิสเก็ตต์จะต้องมีโปรแกรมเหล่านี้อยู่คือ

COBLIB.REL 23 K

COBOL.COM 31 K

LINK.COM 8 K

OVRLY1.OVR 7 K

OVRLY2.OVR 11 K

OVRLY3.OVR 11 K

OVRLY4.OVR 2 K

รูปแบบในการคอมพลายจะต้องกระทำตามรูปฟอร์มคำสั่งต่อไปนี้

1. COBOL XXX,TTY · = XXX.COB
2. COBOL = XXX
3. COBOL ,TTY: = XXX
4. COBOL XXX = XXX.COB
5. COBOL XXX,XXX = XXX
6. COBOL , = XXX

ในแบบที่ 1 จะได้อ็อบเจ็กต์โปรแกรม เป็น REL และแสดงการคอมพลายทางจอ CRT

ในแบบที่ 2 จะได้อ็อบเจ็กต์โปรแกรม เป็น REL และโปรแกรมซอร์สไฟล์จะได้รับการแทนที่ด้วย REL

ในแบบที่ 3 จะได้อ็อบเจ็กต์โปรแกรมออกทาง CRT แต่ไม่สร้าง REL ไฟล์

ในแบบที่ 4 จะได้อ็อบเจ็กต์โปรแกรมมี EXT เป็น REL

ในแบบที่ 5 จะได้อ็อบเจ็กต์โปรแกรมมี EXT เป็น REL และโปรแกรมลิสต์มี EXT เป็น PRN

ในแบบที่ 6 จะไม่มีอ็อบเจ็กต์โปรแกรมออกมา

ในการคอมพลายจะมีสวิตช์ (/) อยู่ด้วยกัน 3 ตัวคือ

/ R หมายถึงการเขียนอ็อบเจ็กต์โปรแกรมกลับเข้าไปเป็นไฟล์ข้อมูลในแผ่นดิสเก็ตต์ (ปกติไม่ใช้)

/ L หมายถึงการสร้างลิสติ้ง ที่มีชนิดของไฟล์ข้อมูลเป็น PRN ในดิสเก็ตต์

/ P หมายถึงการเตรียมที่ไว้ 100 ไบท์ ในระหว่างการคอมพลายซึ่งปกติแล้วห้ามใช้ จะใช้ในกรณีแสตคเกิดโอเวอร์ไฟล์ขึ้นเท่านั้น เมื่อทำการคอมพลายเสร็จแล้วจะออกสู่โปรแกรมจัดระบบงาน

การพิมพ์ผลลัพธ์หลังการคอมพลาย

การคอมพลายของโคบอลจะทำแบบบรรทัดต่อบรรทัด คือ จะมีข้อผิดพลาดบางส่วนแสดงให้เห็นบรรทัดต่อบรรทัด บางส่วนจะแสดงให้เห็นเมื่อคอมพลายเสร็จ

ปกติบรรทัดของเดิมจะประกอบด้วยตัวเลขฐานสิบ 4 หลัก เมื่อเกิดข้อผิดพลาดขึ้นแล้วจะชี้ตัวเลขบรรทัดที่เกิดข้อผิดพลาดนั้น

การเอ็กซีคิว

หลังจากการคอมพลายแล้ว จะใช้โปรแกรม LINK เพื่อทำการโหลดโปรแกรมที่คอมพลายแล้วลงในหน่วยความจำแล้วทำการเอ็กซีคิว (ถ้าต้องการ)

คำสั่งโดยทั่วไปสำหรับ LINK.COM จะประกอบด้วยชื่อไฟล์และสวิตช์ซึ่งมีรูปแบบดังนี้

LINK d: filename 1. ext/S, d: filename 2. ext/S, ...

เมื่อ d คือดิสค์ไดร์ฟที่มีโปรแกรมอยู่

s คือ สวิตช์

ในกรณีที่ไม่เปลี่ยนดิสค์ไดร์ฟคอมพิวเตอร์จะทราบเองว่าเป็นดิสค์ไดร์ฟที่กำลังทำการติดต่ออยู่คือไดร์ฟใด

สวิตช์ที่ใช้ในการลิ้งค์ปกติจะตามด้วย (/) และตัวอักษรต่อไปนี้คือ

1. E(EXIT TO OS) หมายความว่าให้ทำการลิ้งค์เมื่อเสร็จแล้วให้ออกสู่ OS ก่อนออก จะมีตัวเลขปรากฏ มีลักษณะดังนี้

(XXXX YYYY ZZ)

XXXX เป็นค่าแอดเดรสเริ่มต้นในการเอ็กซีคิวชั่น

YYYY เป็นค่าแอดเดรสสุดท้ายในการเอ็กซีคิวชั่น

ZZ เป็นจำนวนหน้าทั้งหมดที่ใช้

จากนั้นก็จะออกสู่โปรแกรมจัดระบบงานแล้วจะต้องทำการ save เพื่อทำให้เป็น .COM file โดยมีลักษณะคำสั่งดังนี้

SAVE filename.COM ZZ

2. G(GO-STARTEXECUTION) มีลักษณะเหมือนกับ E ทุกประการ แต่แทนที่จะออกสู่โปรแกรมจัดระบบงานแต่จะทำการ RUN โปรแกรม ซึ่งจะมีตัวอักษรบอกว่า

(begin execution)

และจะสร้างไฟล์ .COM ให้ด้วย

3. M(MAP ALL SYMBOL) เป็นการแสดงสัญลักษณ์ต่าง ๆ ที่มีประโยชน์ในการแก้ไข (debug) โปรแกรม พร้อมทั้งแสดงแอดเดรสของสัญลักษณ์เหล่านั้นด้วย

4. R(RESET LINKER) จะใช้เมื่อเรียกไฟล์ผิดอันมา โดยจะกลับไปที่จุดเริ่มต้นใหม่

5. S(SEARCH FILE) ใช้สำหรับค้นหาชื่อไฟล์ที่ต้องการ

6. U(List all Undefind global) ให้มีการแสดงสัญลักษณ์ที่ไม่ได้กำหนดไว้อีกครั้งหนึ่ง และสามารถพิมพ์ทางเครื่องพิมพ์ได้ โดยใช้คีย์ ^P ตามด้วย /U

ในการใช้คำสั่ง LINK อาจจะเกิดข้อผิดพลาดได้หลายประการถ้าขึ้นเครื่องหมาย % โปรแกรมอาจจะทำงานได้ในบางส่วน เครื่องหมาย ? จะต้องทำการลิงค์ใหม่

ภาคผนวก ข.

ตัวอย่างการเขียนโปรแกรม

ตัวอย่างการเขียนโปรแกรม

1. การเขียนโปรแกรมด้วยภาษา c-basic ให้ทำการคำนวณหาพื้นที่ของรูปสามเหลี่ยมโดยให้ input เป็นความยาวของด้านทั้ง 3 ของรูปสามเหลี่ยม จะมีวิธีรวมทั้งขั้นตอนในการเขียนและส่งโปรแกรม compile และ RUN ดังนี้ เมื่อให้คำที่ขีดเส้นใต้เป็นคำที่ทำการเขียน

```
A>ED CYO.BAS                          — ทำการเรียกไฟล์ ED และตั้งชื่อไฟล์เป็น CYO
                                        BAS
NEW FILE
*VI                                   — คำสั่งแรกที่ใช้เพื่อจัดบรรทัดสำหรับการเขียนโปรแกรม
  1:  10 INPUT A,B,C
  2:  20 S=(A+B+C)/2
  3:  30 AREA=SQR(S*(S-A)*(S-B)*(S-C))   — ตัวโปรแกรม
  4:  40 OUT AREA
  5:  50 END
  6:  ^Z                              — การสิ้นสุดการเขียนโปรแกรม
   : *E          — การออกจากไฟล์ ED

A>CBAS2 CYO                           — การส่งคอมพลาย
CBASIC COMPILER VER 2.01
   1* 10 INPUT A,B,C
   2* 20 S=(A+B+C)/2
   3* 30 AREA=SQR(S*(S-A)*(S-B)*(S-C))
   4* 40 OUT AREA
ERROR SE IN LINE    4 AT POSITION   12    -ERRORที่เกิดในโปรแกรมที่บรรทัดที่ 4
   5* 50 END
   1 ERROR DETECTED                   — การคอมพลายไม่สำเร็จ
CONSTANT AREA:        8
CODE SIZE:           59
DATA STMT AREA:       0
VARIABLE AREA:       40
A>ED CYO.BAS                          — เรียกไฟล์ ED ใหม่เพื่อทำการแก้ไข
*#AV     — คำสั่งแรกที่ใช้เมื่อทำการแก้ไขโปรแกรมใหม่
   1: *10T                            — list โปรแกรมขึ้นมา 10 บรรทัด
```

```
    1:  10 INPUT A,B,C
    2:  20 S=(A+B+C)/2
    3:  30 AREA=SQR(S*(S-A)*(S-B)*(S-C))     — ตัวโปรแกรม
    4:  40 OUT AREA
    5:  50 END
    1: *4:KI                                  — ทำการลบบรรทัดที่ 4 ทิ้งและทำการเติมโปรแกรมเข้า
    4:  40 PRINT A,B,C,S                        ไปใหม่อีก 2 บรรทัดคือบรรทัดที่ 40 และ 45
    5:  45 PRINT "THE AREA IS ",AREA
    6:  ^Z                                    — สิ้นสุดการเขียนโปรแกรม
    6: *B10T                                  — ทำการ list โปรแกรม 10 บรรทัดมาดูบนจอโทร
    1:  10 INPUT A,B,C                          ทัศน์ โดยเริ่มจากต้นโปรแกรม
    2:  20 S=(A+B+C)/2
    3:  30 AREA=SQR(S*(S-A)*(S-B)*(S-C))
    4:  40 PRINT A,B,C,S
    5:  45 PRINT "THE AREA IS ",AREA
    6:  50 END
    1: *E                                     — ออกจากไฟล์ ED

A>CBAS2 CYO                                   — ทำการส่งคอมพลายใหม่
CBASIC COMPILER VER 2.01

    1* 10 INPUT A,B,C
    2* 20 S=(A+B+C)/2
    3* 30 AREA=SQR(S*(S-A)*(S-B)*(S-C))
    4* 40 PRINT A,B,C,S
    5* 45 PRINT "THE AREA IS ",AREA
    6* 50 END                                 — ไม่มีข้อผิดพลาด
NO ERRORS DETECTED
CONSTANT AREA:       8
CODE SIZE:          95
DATA STMT AREA:      0
VARIABLE AREA:      40
A>CRUN2 CYO                                   — ทำการ RUN โปรแกรม

CRUN VER 2.03

? 3,4,5                                       — อินพุตที่ใส่เข้าไปเป็นด้านทั้งสามของสามเหลี่ยม
 3                    4
THE AREA IS           6                       — เอาต์พุตที่ออกมาคือ A,B,C, S และพื้นที่
```

2. ทำการเขียนโปรแกรมด้วยภาษาแอสเซมบลี้ให้เอาต์พุตที่เป็น LED ที่ port FF บนแผงหน้าปัทม์ ทำการชิฟท์ไปทีละบิทจนครบทั้ง 8 ตัวแล้วกลับมาเริ่มต้นใหม่ จะมีขั้นตอนในการทำงานดังต่อไปนี้

```
ED YO1.ASM                                    — เรียกไฟล์ ED และกำหนดไฟล์ใหม่ชื่อ YO1 ASM

NEW FILE
*VI                                           — คำสั่งแรกที่ใช้
    1: ^I       ORG ^I   100H                 — กดปุ่ม control I เพื่อให้เคอร์เซอร์เลื่อนไป 8 ตัว
```

```
  2:  MVVM     MVI      A,1
  3:           CMA
  4:  BEGIN    OUT      OFFH
  5:           RLV
  6:           CALL     RELA
  7:           JMP      BEGIN
  8:  RELA     MVI      B,OFFH
  9:           DCR      B
 10:           RZ
 11:           MVI      C,FF
 12:  T2       DCR      C
 13:           JNZ      T2
 14:           JMP      T1
 15:  ^Z
   :  *E
```

อักษรสองตัวหลังจะเป็นเอโค่ของการลบและจะไม่ปรากฏในตัวโปรแกรม

— การสิ้นสุดการเขียนโปรแกรม

— การออกจากไฟล์ ED.

```
ASM Y01
CP/M ASSEMBLER - VER 1.4
U010F 0E00                    MVI    C,FF
U0115 C30000                  JMP    T1
0118
000H USE FACTOR
END OF ASSEMBLY
A>ED Y01.ASM
*#AV
   1: *20T
   1:           ORG      100H
   2:           MVI      A,1
   3:           CMA
   4:  BEGIN    OUT      OFFH
   5:           RLV
   6:           CALL     RELA
   7:           JMP      BEGIN
   8:  RELA     MVI      B,OFFH
   9:           DCR      B
  10:           RZ
  11:           MVI      C,FF
  12:  T2       DCR      C
  13:           JNZ      T2
  14:           JMP      T1
   1: *5:KI
   5:           RLC
   6:  Z
   6: *9:KI
   9:  T1       DCRBB    B
  10:  ^Z
  10: *11:SFFOFFH
```

— การส่งคอมพลายภาษาแอสเซมบลี้

ข้อผิดพลาดที่เกิดขึ้น (U010F, U0115)

— การเรียกไฟล์ ED มาใหม่เพื่อทำการแก้ไขโปร แกรม

— คำสั่งแรกที่ต้องใช้เมื่อเรียกไฟล์ ED มาทำการ

— ทำการเรียกโปรแกรมมาดู 20 บรรทัด แก้ไข

— ตัวโปรแกรม

— ทำการลบบรรทัดที่ 5 ทิ้งแล้วทำการเขียนเติมลงไปแทนด้วย RIC

—สิ้นสุดการเขียนโปรแกรม

— ลบบรรทัดที่ 9 ทิ้งและเติมลงไปดังข้างล่างนี้

— ในบรรทัดที่ 11 ให้ทำการหาคำว่า FF แล้วเปลี่ยน

```
  11: *B20T
   1:          ORG      100H
   2:          MVI      A,1
   3:          CMA
   4:  BEGIN   OUT      OFFH
   5:          RLC
   6:          CALL     RELA
   7:          JMP      BEGIN
   8:  RELA    MVI      B,OFFH
   9:  T1      DCR      B
  10:          RZ
  11:          MVI      C,OFFH
  12:  T2      DCR      C
  13:          JNZ      T2
  14:          JMP      T1
   1: *E
```

— เป็น OFFH และในบรรทัดต่อมาจะสั่งให้เรียกโปรแกรมมาดู 20 บรรทัดโดยเริ่มจากบรรทัดแรก

— ออกจากไฟล์ ED

```
A>ASM YO1
CP/M ASSEMBLER - VER 1.4
0119
000H USE FACTOR
END OF ASSEMBLY
```

— คอมพลายไฟล์ YO1. อีกครั้ง

— ไม่มีข้อผิดพลาด

```
A>LOAD YO1
```

— ทำการ LOAD เพื่อสร้างคำสั่งทรานเชียนต์

```
FIRST ADDRESS 0100
LAST ADDRESS 0118
BYTES READ    0019
RECORDS WRITTEN 01

A>YO1
```

— คำสั่งให้ทำงานแบบคำสั่งทรานเชียนต์

3. ตัวอย่างการเขียนโปรแกรมด้วยภาษาเบสิคของไมโครซอฟต์ให้ทำการคำนวณเปลี่ยนมาตราของอุณหภูมิจากฟาเรนไฮท์ไปเป็น เซนติเกรด ดังนี้ โดยเริ่มแรกต้องทำการเรียก **MBASIC** ลงสู่หน่วยความจำเสียก่อนดังนี้

```
A>mbasic
26535 Bytes free
BASIC Rev. 4.45
[CP/M Version]
Copyright 1977 (C) by Microsoft
Ok
AUTO
10 PRINT "FAHRENHEIT"
20 INPUT F
```

— คำสั่งที่ให้จัดบรรทัดโดยเริ่มต้นที่ 10 และเพิ่มขึ้นทีละ 10

— ตัวโปรแกรม

```
30 LETCC(F-32)*5/9                ] ตัวโปรแกรม
40 PRINT "CENTIGRADE",C
50 END
60 ^C                             — การสิ้นสุดการเขียนโปรแกรม
Ok
RUN                               — ทำการ RUN โปรแกรมนี้
FAHRENHEIT
? 100                             — อินพุตที่เป็นฟาเรนไฮท์
Syntax error in 30
Ok                                — โปรแกรมนี้มีข้อผิดพลาดที่บรรทัดที่ 30
30
DIT
Syntax error                      — ทำการแก้ไขโปรแกรมที่บรรทัดที่ 30 โดยสั่งด้วย
Ok                                  คำสั่งว่า EDIT 30 แต่ถ้าใช้คำสั่งที่ผิดเครื่องจะ
EDIT                                ทำการบอกให้รู้ดังตัวอย่างเมื่อเขียนเป็น DIT
Undefined line number               หรือ EDIT
Ok
EDIT 30                           — ตัวอย่างที่ถูก
30↑LETCC(F-32)*5/9                — เครื่องหมาย ↑ ในที่นี้จะหมายถึง carriage return
30 LET C=(F-32)*5/9
^C                                — แล้วในบรรทัดต่อไปก็ทำการแก้ไขให้ถูกแล้วจบด้วย ∧ C
RUN                                 ซึ่งจะเป็นคำสั่งที่บอกให้เครื่องรู้ว่าสิ้นสุดการแก้ไขแล้ว
FAHRENHEIT
? 100                             — ทำการ RUN ซึ่งเครื่องจะถามหาอินพุต
CENTIGRADE      37.7778           — เมื่อทำการคำนวณแล้วจะได้เอาต์พุตออกมา
Ok
SAVE "MEX1"                       — ทำการเก็บโปรแกรมนี้ลงสู่แผ่นดิสเก็ตต์แล้วตั้งชื่อ
Ok                                  ให้ว่า MEX 1
LOAD "MEX1"                       — ทำการเรียกไฟล์ MEX 1 จากแผ่นดิสเก็ตต์
Ok
LIST                              — ลงสู่หน่วยความจำ แล้วใช้คำสั่งเรียกเพื่อดูตัวโปรแกรม
10 PRINT "FAHRENHEIT"
20 INPUT F
30 LET C=(F-32)*5/9
40 PRINT "CENTIGRADE",C
50 END
Ok
SYSTEM                            — เป็นคำสั่งที่สั่งเมื่อต้องการออกจาก MBASIC
                                    สู่ CP/M
A>
```

4. ตัวอย่างการใช้ DDT ในการแก้ไขโปรแกรมโดยไม่ต้องเข้าไปแก้ไขในไฟล์ ED และยังสามารถตรวจสอบโปรแกรมในขณะนั้นได้ทันที มีวิธีการใช้ดังนี้

```
TYPE YO2.PRN                      — การเรียกตัวโปรแกรมขึ้นดูบนจอโทรทัศน์ก่อน
                                    เรียกเข้าสู่ DDT
 F154 =        SETDSK  EQU     0F154H
```

```
F197 =          SETTRK  EQU     0F197H
F19C =          SETSEC  EQU     0F19CH
F1A1 =          SETDMA  EQU     0F1A1H
F221 =          READ    EQU     0F221H
0100                    ORG     100H
0100 010080             LXI     B,8000H ;  SET DMA ADDRESS.
0103 CDA1F1             CALL    SETDMA
0106 0E00               MVI     C,00H   ;  SET DISK TO DRIVE A.
0108 CD54F1             CALL    SETDSK
010B 0E02               MVI     C,02H   ;  SET TRACK TO TRACK 2
010D CD97F1             CALL    SETTRK
0110 0E03               MVI     C,03H   ;  SET SECTOR TO SECTOR 3.
0112 CD9CF1             CALL    SETSEC
0115 CD21F2             CALL    READ    ;  START TO READ DATA FROM
                ;DRIVE A,TRACK 2,SECTOR 3 TO ADDRESS 8000H.
0118 C30000             JMP     0000H
```

```
A>DDT YO2.COM                          — เรียกไฟล์ YO2.COM เข้าสู่ DDT
DDT VERS 1.4
NEXT  PC
0180 0100                              — โปรแกรมจะเริ่มจากแอดเดรส 100 ถึง 180
-L                                     — ทำการเรียกโปรแกรมมาดู 11 บรรทัด
  0100  LXI  B,8000
  0103  CALL F1A1
  0106  MVI  C,00
  0108  CALL F154
  010B  MVI  C,02
  010D  CALL F197
  0110  MVI  C,03
  0112  CALL F19C
  0115  CALL F221
  0118  JMP  0000
  011B  NOP
-M100,11B,9000                         — ทำการย้ายข้อมูล (ตัวโปรแกรม) ตั้งแต่เอดเดรส /100/
-L9000                                   ถึง 11B ไปไว้ที่แอดเดรสใหม่ที่ 9000
  9000  LXI  B,8000                    — แล้วทำการ list โปรแกรมที่แอดเดรสใหม่ที่ 9000
  9003  CALL F1A1                        ขึ้นมาดูบนจอโทรทัศน์
  9006  MVI  C,00
  9008  CALL F154
  900B  MVI  C,02
  900D  CALL F197
  9010  MVI  C,03
  9012  CALL F19C
  9015  CALL F221
  9018  JMP  0000
                                       — ทำการเติมข้อมูลลงไปยังหน่วยความจำตั้งแต่แอด
-F8000,8080,FF                           เดรส 8000 ถึงที่ 8080 ด้วยค่าคงที่คือ FF แล้วทำ
```

-D8000 — การ dump ดูค่าที่อยู่ภายในหน่วยความจำตั้งแต่แอดเดรส 8000 ไปขนาด 12 × 16 ตำแหน่ง

```
8000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
8010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
8020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
8030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
8040 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
8050 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
8060 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
8070 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
8080 FF 01 20 00 01 01 00 00 00 01 00 01 01 01 00 00 .. .............
8090 FE FE FF FE FF FE FE FE FE FE FE FE FF FF FF FF ................
80A0 01 01 00 40 01 01 00 00 01 01 01 01 01 01 01 00 ...@............
80B0 FE FE FE FF FF FF FE FE FE FF FF FE FE FF FE FE ................
-G9000,9018
*9018
-
```

— ทำการ RUN โปรแกรมโดยให้เริ่มตั้งแต่แอดเดรสที่ 9000 ไปจนถึงแอดเดรสที่ 9018

— ทำการ dump ค่าในหน่วยความจำตั้งแต่แอดเดรสที่ 8000 มาดูใหม่ ซึ่งจะได้เป็นข้อมูลของโปรแกรมที่ได้และเป็นข้อมูลที่อยู่ภายในแผ่นดิสเก็ตต์ที่แทร็ค 2 เซกเตอร์ 3

```
-D8000
8000 00 4B 55 32 20 20 20 20 20 41 53 4D 00 65 00 32 .KU2     ASM.e.2
8010 A3 A4 A5 BA BB BC BE 00 00 00 00 00 00 00 00 00 ................
8020 00 4B 55 32 20 20 20 20 20 42 41 4B 00 5A 00 30 .KU2     BAK.Z.0
8030 A0 A1 A2 A9 AA AB 00 00 00 00 00 00 00 00 00 00 ................
8040 00 59 4F 31 20 20 20 20 20 41 53 4D 00 6E 00 02 .YO1     ASM.n..
8050 BD 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
8060 00 59 4F 31 20 20 20 20 20 42 41 4B 00 72 00 02 .YO1     BAK.r..
8070 A6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
8080 FF 01 20 00 01 01 00 00 00 01 00 01 01 01 00 00 .. .............
8090 FE FE FF FE FF FE FE FE FE FE FE FE FF FF FF FF ................
80A0 01 01 00 40 01 01 00 00 01 01 01 01 01 01 01 00 ...@............
80B0 FE FE FE FF FF FF FE FE FE FF FF FE FE FF FE FE ................

-IYO1.COM
-R
NEXT  PC
0180 0100
-L100
```

— ทำการเรียกไฟล์ YO1.COM เข้าสู่หน่วยความจำใหม่อีก

— โดยจะเริ่มอยู่ที่แอดเดรส 100 ถึง 180

— ทำการเรียกโปรแกรมมาดู

```
  0100   MVI   A,01
  0102   CMA
  0103   OUT   FF
  0105   RLC
  0106   CALL  010C
  0109   JMP   0103
  010C   MVI   B,FF
  010E   DCR   B
  010F   RZ
  0110   MVI   C,FF
  0112   DCR   C
-L
```

— ทำการเรียกโปรแกรมต่อไปมาดู

```
 0113  JNZ  0112
0116  JMP  010E

I 0119  NOP
 011A  NOP
 011B  NOP
 011C  NOP
 011D  NOP
 011E  NOP
 011F  NOP
 0120  NOP
 0121  NOP
 0122  NOP
 0123  NOP
 0124
-X
```

— ทำการดูค่าข้อมูลของรีจิสเตอร์ต่าง ๆ

```
C0Z1M0E1I1 A=00 B=0002 D=000A H=8080 S=0100 P=0118 JMP  0000
-XB
```

— ทำการเซตค่าภายในรีจิสเตอร์ BC ให้เป็น 00

```
B=0002 0000
-XP
```

— ทำการเซตค่าโปรแกรมเคาวน์เตอร์ไปไว้ที่ 100

```
P=0118 100
-X
```

— ดูค่าภายในรีจิสเตอร์ต่าง ๆ ใหม่

```
C0Z1M0E1I1 A=00 B=0000 D=000A H=8080 S=0100 P=0100 LXI  B,8000
-T15
```

—ทำการทดลองโปรแกรมไป 21 สเต็ป (15_{16})

```
C0Z1M0E1I1 A=00 B=0000 D=000A H=8080 S=0100 P=0100 MVI  A,01
C0Z1M0E1I1 A=01 B=0000 D=000A H=8080 S=0100 P=0102 CMA
C0Z1M0E1I1 A=FE B=0000 D=000A H=8080 S=0100 P=0103 OUT  FF
C0Z1M0E1I1 A=FE B=0000 D=000A H=8080 S=0100 P=0105 RLC
C1Z1M0E1I0 A=FD B=0000 D=000A H=8080 S=0100 P=0106 CALL 010C
C1Z1M0E1I0 A=FD B=0000 D=000A H=8080 S=00FE P=010C MVI  B,FF
C1Z1M0E1I0 A=FD B=FF00 D=000A H=8080 S=00FE P=010E DCR  B
C1Z0M1E0I0 A=FD B=FE00 D=000A H=8080 S=00FE P=010F RZ
C1Z0M1E0I0 A=FD B=FE00 D=000A H=8080 S=00FE P=0110 MVI  C,FF
C1Z0M1E0I0 A=FD B=FEFF D=000A H=8080 S=00FE P=0112 DCR  C
C1Z0M1E0I0 A=FD B=FEFE D=000A H=8080 S=00FE P=0113 JNZ  0112
C1Z0M1E0I0 A=FD B=FEFE D=000A H=8080 S=00FE P=0112 DCR  C
C1Z0M1E0I0 A=FD B=FEFD D=000A H=8080 S=00FE P=0113 JNZ  0112
C1Z0M1E0I0 A=FD B=FEFD D=000A H=8080 S=00FE P=0112 DCR  C
C1Z0M1E0I0 A=FD B=FEFC D=000A H=8080 S=00FE P=0113 JNZ  0112
C1Z0M1E0I0 A=FD B=FEFC D=000A H=8080 S=00FE P=0112 DCR  C
C1Z0M1E0I0 A=FD B=FEFB D=000A H=8080 S=00FE P=0113 JNZ  0112
C1Z0M1E0I0 A=FD B=FEFB D=000A H=8080 S=00FE P=0112 DCR  C
C1Z0M1E0I0 A=FD B=FEFA D=000A H=8080 S=00FE P=0113 JNZ  0112
C1Z0M1E0I0 A=FD B=FEFA D=000A H=8080 S=00FE P=0112 DCR  C
C1Z0M1E0I0 A=FD B=FEF9 D=000A H=8080 S=00FE P=0113 JNZ  0112*0112
-L100
```

— เรียกโปรแกรมมาดู 11 สเต็ป

```
  0100  MVI  A,01
  0102  CMA
```

```
0103  OUT  FF
0105  RLC
0106  CALL 010C
0109  JMP  0103
010C  MVI  B,FF
010E  DCR  B
010F  RZ
0110  MVI  C,FF
0112  DCR  C
```

```
-A10C                        — ทำการเขียนโปรแกรมลงไปใหม่ที่แอดเดรส 10C
010C  MVI B,02
010E  cr                     — สิ้นสุดการเขียนโปรแกรม
-S110                        — ทำการเติมค่าคงที่ลงไปที่แอดเดรส 110 เป็นต้นไป
0110 0E                        โดยที่แอดเดรส 110 จะไม่เปลี่ยน ที่ 111 จะเปลี่ยน
0111 FF 02                     จาก FF เป็น 2 และใช้เครื่องหมาย . เป็นสัญ
0112 0D _                      ลักษณ์ในการออกจากหรือสิ้นสุดการเติม
-
-L100                        — ทำการ list โปรแกรมมาดู โดยเริ่มจากแอด
 0100  MVI  A,01               เดรสที่ 100
 0102  CMA
 0103  OUT  FF
 0105  RLC
 0106  CALL 010C
 0109  JMP  0103
 010C  MVI  B,02
 010E  DCR  B
 010F  RZ
 0110  MVI  C,02
 0112  DCR  C                — เครื่องหมาย # เป็นเครื่องหมายที่เกิดจากการกด
-T15#                          ปุ่ม ctl-U และเขียนใหม่เป็น X
X
C1Z0M1E0I0 A=FD B=FEF9 D=000A H=8080 S=00FE P=0112 DCR  C
-XP
P=0112 100                   — ทำการเซตค่าโปรแกรมเคาวน์เตอร์เป็น 100
-XM
M1 0                         — เซตค่าแฟล็กลบให้เป็น 0
-T17          —ทดสอบโปรแกรมไป 23 สเต็ป (= 17₁₆)
C1Z0M0E0I0 A=FD B=FEF9 D=000A H=8080 S=00FE P=0100 MVI  A,01
C1Z0M0E0I0 A=01 B=FEF9 D=000A H=8080 S=00FE P=0102 CMA
C1Z0M0E0I1 A=FE B=FEF9 D=000A H=8080 S=00FE P=0103 OUT  FF
C1Z0M0E0I1 A=FE B=FEF9 D=000A H=8080 S=00FE P=0105 RLC
C1Z0M0E0I0 A=FD B=FEF9 D=000A H=8080 S=00FE P=0106 CALL 010C
C1Z0M0E0I0 A=FD B=FEF9 D=000A H=8080 S=00FC P=010C MVI  B,02
C1Z0M0E0I0 A=FD B=02F9 D=000A H=8080 S=00FC P=010E DCR  B
C1Z0M0E0I0 A=FD B=01F9 D=000A H=8080 S=00FC P=010F RZ
C1Z0M0E0I0 A=FD B=01F9 D=000A H=8080 S=00FC P=0110 MVI  C,02
```

```
C1Z0M0E0I0 A=FD B=0102 D=000A H=8080 S=00FC P=0112 DCR  C
C1Z0M0E0I0 A=FD B=0101 D=000A H=8080 S=00FC P=0113 JNZ  0112
C1Z0M0E0I0 A=FD B=0101 D=000A H=8080 S=00FC P=0112 DCR  C
C1Z1M0E0I0 A=FD B=0100 D=000A H=8080 S=00FC P=0113 JNZ  0112
C1Z1M0E0I0 A=FD B=0100 D=000A H=8080 S=00FC P=0116 JMP  010E
C1Z1M0E0I0 A=FD B=0100 D=000A H=8080 S=00FC P=010E DCR  B
C1Z1M0E0I0 A=FD B=0000 D=000A H=8080 S=00FC P=010F RZ
C1Z1M0E0I0 A=FD B=0000 D=000A H=8080 S=00FE P=0109 JMP  0103
C1Z1M0E0I0 A=FD B=0000 D=000A H=8080 S=00FE P=0103 OUT  FF
C1Z1M0E0I0 A=FD B=0000 D=000A H=8080 S=00FE P=0105 RLC
C1Z1M0E0I0 A=FB B=0000 D=000A H=8080 S=00FE P=0106 CALL 010C
C1Z1M0E0I0 A=FB B=0000 D=000A H=8080 S=00FC P=010C MVI  B,02
C1Z1M0E0I0 A=FB B=0200 D=000A H=8080 S=00FC P=010E DCR  B
C1Z0M0E0I0 A=FB B=0100 D=000A H=8080 S=00FC P=010F RZ   *0110
-G0          - ออกจาก DDT
```

ภาคผนวก ค.

การจัดไฟล์ข้อมูลในแผ่นดิสเก็ตต์

ในแผ่นดิสเก็ตต์ที่ใช้กันกับเครื่องไมโครคอมพิวเตอร์ ส่วนใหญ่จะเป็นชนิดซอฟต์เซกเตอร์เสียเป็นส่วนใหญ่ ซึ่งจะมีลักษณะเป็นแผ่นกลมและแบน ถูกฉาบด้วยสารแม่เหล็กบนบริเวณผิวหน้าของแผ่นดิสเก็ตต์ ถ้าในการฉาบสารแม่เหล็กนี้ถูกฉาบเพียงด้านเดียวจะเรียกว่าเป็นชนิด single-side แต่ถ้าถูกฉาบด้วยสารแม่เหล็กทั้งสองด้านจะเรียกว่า double-side แผ่นดิสเก็ตต์นี้จะถูกป้องกันโดยซองพลาสติกสี่เหลี่ยมที่จะเจาะช่องไว้สำหรับหัวอ่านของเครื่องอ่านดิสเก็ตต์ใช้สำหรับการอ่านและการเขียนลงยังแผ่นดิสเก็ตต์ และตรงบริเวณตอนกลางของแผ่นดิสเก็ตต์จะเจาะไว้เป็นรูกลมเพื่อใช้ในการจับแผ่นดิสเก็ตต์หมุน ภายในซองพลาสติกนี้จะมีลักษณะคล้ายผ้าเพื่อใช้ถูทำความสะอาดดิสเก็ตต์ในขณะที่แผ่นดิสเก็ตต์หมุน แผ่นดิสเก็ตต์ที่ใช้กันอยู่มีอยู่ด้วยกัน 2 ขนาดคือขนาด 8 นิ้ว และ 5 นิ้ว ซึ่งมีลักษณะการจัดรูปแบบของการเก็บไฟล์ข้อมูลตามแบบของ IBM เป็นดังนี้ คือ

ในดิสค์ขนาด 8 นิ้วจะมีความจุ 241 กิโลไบท์ จะแบ่งออกเป็น 77 แทร็ค (track 0-track 76) ใน 1 แทร็ค จะแบ่งออกเป็น 26 เซกเตอร์ (sector 1 - sector 26)

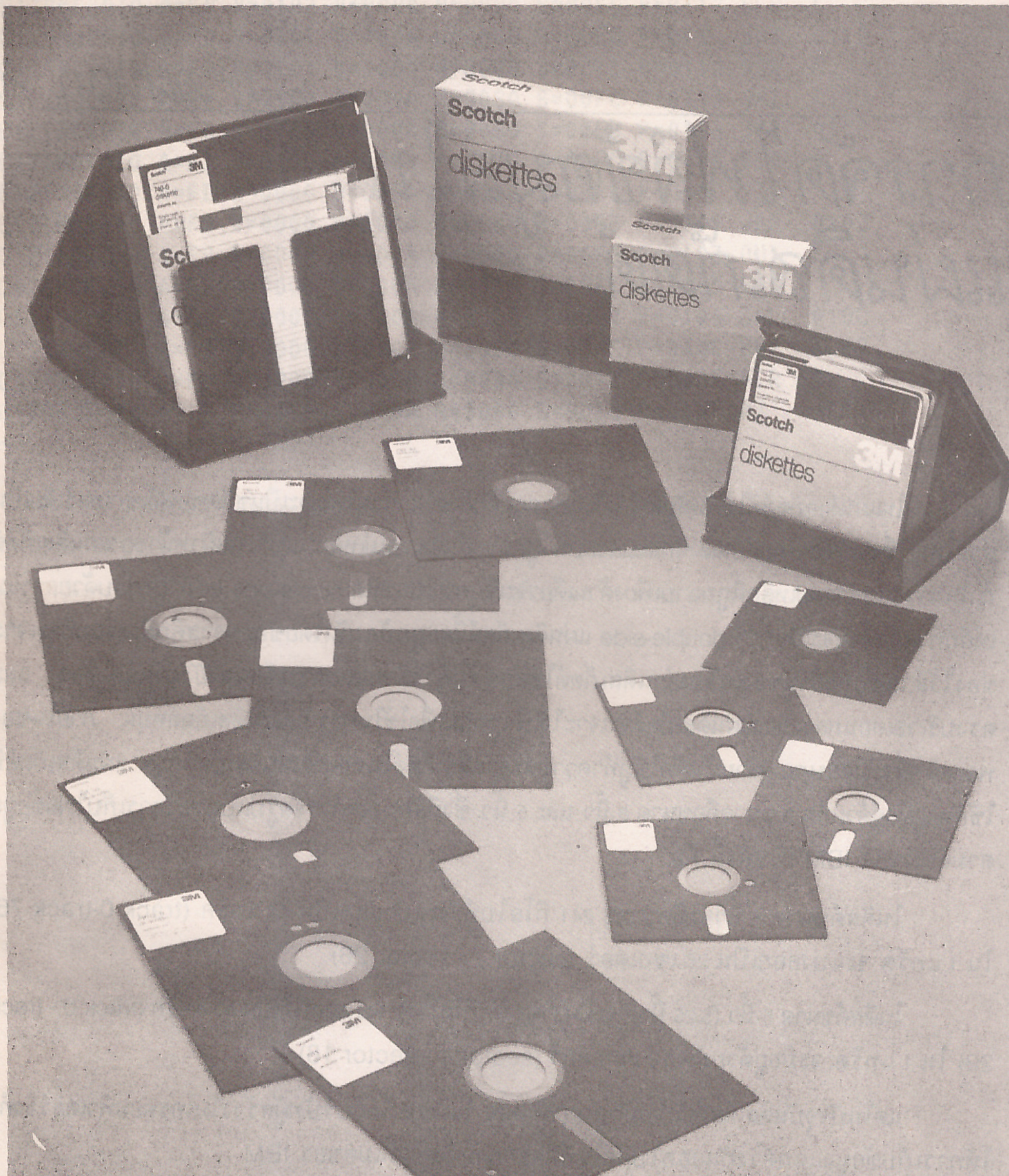
ในดิสค์ขนาด 5 นิ้ว (5.25 นิ้ว) จะมีความจุ 81 กิโลไบท์ แบ่งออกได้เป็น 40 แทร็ค (track 0 - track 39) ใน 1 แทร็ค จะมีอยู่ด้วยกัน 18 เซกเตอร์ (sector 1 - sector 18)

แต่ในปัจจุบันดิสเก็ตต์ทั่วไปมักถูกกำหนดฟอร์แมตเองโดยบริษัทผู้สร้างไมโครคอมพิวเตอร์จึงทำให้การเก็บข้อมูลแตกต่างกันออกไปแต่ในที่นี้จะกล่าวถึงฟอร์แมตของ IBM

การแบ่งส่วนในแผ่นฟลอปปี้ดิสค์

ในแผ่นดิสเก็ตต์ชนิด single-side ขนาด 8 นิ้วจะแบ่งโครงสร้างออกเป็น 4 ส่วนดังนี้คือ

– บูทสแทรปโหลดเดอร์
– โปรแกรมระบบปฏิบัติงาน
– สารบัญไฟล์
– ไฟล์ใช้งาน



รูปที่ 1 ตัวอย่างแผ่นดิสเก็ตต์ขนาดต่าง ๆ

ตำแหน่งของส่วนต่าง ๆ เหล่านี้จะสามารถแสดงได้ดังรูปที่ 2

ไฟล์ใช้งาน	แทร็ค 2 - แทร็ค 7
สารบัญไฟล์	แทร็ค 2
โปรแกรมระบบปฏิบัติงาน	แทร็ค 0 - แทร็ค 1
บูทสแทรปโหลดเดอร์	แทร็ค 0 เซกเตอร์ 1

รูปที่ 2 แสดงตำแหน่งของโครงสร้างของแผ่นดิสเก็ตต์ขนาด 8 นิ้ว

บูทสแทรปโหลดเดอร์ เป็นส่วนที่อยู่ภายในแผ่นดิสค์ที่แทร็คที่ศูนย์เซกเตอร์ที่ 1 โดยทั่วไปเมื่อทำการเปิดเครื่องและทำการบูทโปรแกรมเครื่องไมโครคอมพิวเตอร์จะทำการรันโปรแกรมในส่วนของ ROM หรือมอนิเตอร์เพื่อทำการอ่านเอาส่วนของบูทสแทรปโหลดเดอร์จากดิสเก็ตต์ลงสู่หน่วยความจำที่แอดเดรส 0000 (เลขฐานสิบหก) เป็นต้นไป แล้วจะทำการย้ายโปรแกรมเคาวน์เตอร์ไปชี้ที่แอดเดรส 0000 (เลขฐานสิบหก) และจะเริ่มต้น RUN ที่นี่ ซึ่งจะเป็นโปรแกรมที่จะทำการอ่านข้อมูลจากแผ่นดิสค์ในส่วนที่เป็นโปรแกรมระบบปฏิบัติงาน (OS.) ไปไว้ในหน่วยความจำยังตำแหน่งที่ได้ไบแอสไว้แล้ว

โปรแกรมระบบปฏิบัติงาน (OS.) เป็นส่วนของโปรแกรมที่ใช้ในการบังคับและควบคุมการทำงานของเครื่องไมโครคอมพิวเตอร์ ซึ่งส่วนของโปรแกรมเหล่านี้จะอยู่ภายในแผ่นดิสค์ที่แทร็คที่ 0 (ยกเว้นใน sector แรก) และในแทร็คที่ 1 ตัวอย่างของตำแหน่งของส่วนต่าง ๆ ของ CP/M ที่อยู่บนเครื่องอิมไซ

ในส่วนของ CCP จะเริ่มจากแทร็คที่ 0 เซกเตอร์ที่ 2 ถึงแทร็คที่ 0 เซกเตอร์ 17

ในส่วนของ BDOS จะเริ่มจากแทร็คที่ 0 เซกเตอร์ที่ 18 ถึงแทร็คที่ 1 เซกเตอร์ 17

ในส่วนของ BIOS จะเริ่มจากแทร็คที่ 1 เซกเตอร์ที่ 18 ถึงแทร็คที่ 1 เซกเตอร์ 21

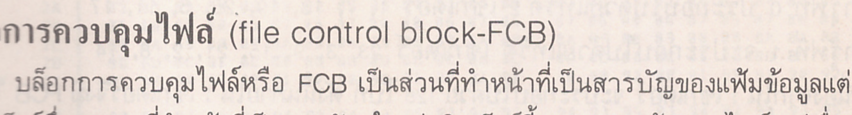
สารบัญดิสค์ไดร์ฟ (disk file directory) เป็นส่วนที่อยู่ภายในแทร็คที่ 2 และเซกเตอร์ที่ 1 2 3 5 7 8 9 11 13 14 15 17 19 21 23 25 และ 26 ซึ่งเซกเตอร์เหล่านี้จะทำหน้าที่ในการเก็บส่วนที่เป็นสารบัญและบล็อกการควบคุมไฟล์ (FCB)

ไฟล์ใช้งาน (user file) เป็นส่วนที่ทำหน้าที่เก็บแฟ้มข้อมูลทั้งหมด โดยเริ่มจากบางส่วนของแทร็คที่ 2 เป็นต้นไป

บล็อกการควบคุมไฟล์ (file control block-FCB)

บล็อกการควบคุมไฟล์หรือ FCB เป็นส่วนที่ทำหน้าที่เป็นสารบัญของแฟ้มข้อมูลแต่ละแฟ้มในแผ่นดิสเก็ตต์ซึ่ง FCB ที่ทำหน้าที่เป็นสารบัญ ในแผ่นดิสเก็ตต์นี้จะประกอบด้วย 32 ไบท์ แต่เมื่อถูกโหลดลงหน่วยความจำ (RAM) และถูกใช้โดย DOS (disk operating system) แล้วจะประกอบด้วย 33 ไบท์ โดยเพิ่มไบท์สุดท้ายขึ้นอีก 1 ไบท์

ลักษณะของรูปแบบของ FCB มีดังรูปที่ 3 ดังนี้



รูปที่ 3 แสดงฟอร์แมต FCB

ความหมายของตำแหน่งต่าง ๆ มีดังต่อไปนี้คือ

สัญลักษณ์	ตำแหน่งที่	ความหมาย
ET	0	เป็นส่วนที่บอกให้รู้ว่า FCB อันนี้ยังคงใช้งานอยู่หรือไม่ โดยถ้าค่าภายในไบท์นี้มีค่าเป็น 00 (เลขฐานสิบหก) จะแสดงว่า FCB ชุดนี้ยังคงใช้งานอยู่ แต่ถ้าค่าภายในไบท์นี้มีค่าเป็น E5H (เลขฐานสิบหก) จะแสดงว่า FCB ชุดนี้ไม่ได้ใช้งานต่อไปอีกแล้ว ซึ่งทั้งนี้อาจจะเป็นเพราะไฟล์ข้อมูลใน FCB นี้ถูกลบทิ้งเสียแล้ว
FN	1 - 8	เป็นส่วนที่ทำหน้าที่ในการเก็บชื่อของไฟล์ข้อมูล (file name) ในรูปของสัญลักษณ์ ASCII ซึ่งมีได้มากที่สุด 8 คำ ในกรณีที่ชื่อของไฟล์ข้อมูลใดมีไม่ครบ 8 คำ ส่วนที่เหลือจะถูกเติมด้วยช่องว่างจนกว่าจะครบ 8 ไบท์
FT	9 - 11	เป็นส่วนที่ทำการเก็บชื่อของชนิดของไฟล์ข้อมูล (file type) ในรูปของสัญลักษณ์ ASCII เช่นเดียวกับ FN
EX	12	เป็นตัวเก็บจำนวน exten ของไฟล์ข้อมูลนั้นว่าในไฟล์ข้อมูลใด ๆ จะมี FCB นี้เป็น exten ที่เท่าใด โดยถ้าเป็น FCB แรกจะมีค่า exten เป็น 00H แต่ถ้าเป็น FCB ที่สองจะมีค่า exten เป็น 01H
	13 - 14	ไม่ได้ถูกใช้ โดยทั่วไปจะมีค่าเป็น 00H
RC	15	เป็นตัวนับจำนวนเซกเตอร์ของไฟล์ข้อมูลนั้นใน FCB ซึ่งจะมีค่าได้ตั้งแต่ 0 ถึง 128
DM	16 - 31	ในแต่ละไบท์แสดงถึงค่าของแต่ละกรุ๊ฟที่ใช้ในการเก็บไฟล์ข้อมูลของ FCB นี้
NR	32	แสดงถึงเรคอร์ดต่อไปที่จะทำการติดต่อในการอ่านหรือเขียนในครั้งต่อไป

คลัสเตอร์ (CLUSTER)

เพื่อสะดวกในการอ่านและเก็บข้อมูลในแผ่นดิสค์ ดังนั้นตั้งแต่แทร็คที่ 2 เป็นต้นไป แผ่นดิสค์จะถูกแบ่งออกเป็นกรุ๊ฟหรือคลัสเตอร์โดยจะเริ่มตั้งแต่กรุ๊ฟที่ 0 เป็นต้นไป จนถึงกรุ๊ฟที่ 242 แต่ละกรุ๊ฟจะประกอบด้วย 8 เซกเตอร์ หรือ 1 กิโลไบท์ และใน CP/M จะมีการจัดให้มีอินเตอร์ลีฟ 6 นั้นคือในแต่ละเซกเตอร์ที่ต่อกันภายในกรุ๊ฟเดียวกันจะต่างกันอยู่ 6 เซกเตอร์ เช่น ในกรุ๊ฟที่ 0 และกรุ๊ฟที่ 1 ซึ่งเป็นส่วนของไดเรกทอรีหรือสารบัญจัดแบ่งคลัสเตอร์ดังนี้คือ

กรุ๊ฟที่ 0 ประกอบไปด้วยแทร็ค 2 เซกเตอร์ 1, 7, 13, 19, 25, 5, 11, 17

กรุ๊ฟที่ 1 จะประกอบไปด้วยแทร็ค เซกเตอร์ 23, 3, 9, 15, 21, 2, 8, 14

เนื่องจากใน 1 เซกเตอร์ จะประกอบไปด้วย 128 ไบท์ ดังนั้นภายใน 1 เซกเตอร์จึงมี FCB ได้เพียง 4 เท่านั้น และเมื่อรวมทั้งกรุ๊ฟที่ 0 และกรุ๊ฟที่ 1 ทั้งหมด ซึ่งมีทั้งหมด 16 เซกเตอร์ จะได้ FCB ทั้งหมดรวม 64 ชุดเท่านั้นซึ่งจะหมายความว่าในแผ่นดิสค์ 1 แผ่นจะมีไฟล์ข้อมูลได้มากที่สุดเพียง 64 ไฟล์ข้อมูลเท่านั้น

เนื่องจากในส่วนของ DM ใน FCB มีเพียง 16 ไบท์เท่านั้น ซึ่งหมายถึงการที่ FCB หนึ่ง ๆ จะสามารถอ้างถึงไฟล์ข้อมูลได้มากที่สุดเป็นขนาดเพียง 16 กิโลไบท์เท่านั้น ถ้าในกรณีที่มีไฟล์ข้อมูลใดมีขนาดเกินกว่า 16 กิโลไบท์ ไฟล์ข้อมูลนั้นจะต้อง FCB มากกว่าหนึ่งชุด โดยที่ FCB ชุดแรกจะมีค่าในส่วนของ EX ไบท์เป็น 00 (เลขฐานสิบหก) และ FCB ต่อมาก็จะมีค่าใน EX ไบท์เป็น 01 (เลขฐานสิบหก), 02 (เลขฐานสิบหก) ,...0E (เลขฐานสิบหก) ตามลำดับ

ในการแบ่งกรุ๊ฟของดิสค์ สามารถค้นหาได้จากตารางที่ 1 ซึ่งจะแสดงถึงการหากรุ๊ฟเมื่อรู้ค่าแทร็คและเซกเตอร์ใดบ้างที่อยู่ภายในกรุ๊ฟที่กำหนดให้

ตารางที่ 1 คลัสเตอร์สำหรับดิสเก็ตต์ขนาด 8 นิ้ว

เซกเตอร์

แทร็ค	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
02	00	01	01	02	00	02	00	01	01	02	00	02	00	01	01	03	00	02	00	02	01	03	01	02	00	02
03	03	04	04	06	03	05	03	05	04	06	04	05	03	05	04	06	04	05	03	05	04	06	04	05	03	05
04	06	08	07	09	07	08	06	08	07	09	07	08	06	08	07	09	07	09	06	08	08	09	07	09	07	08
05	09	0B	0A	0C	0A	0C	09	0B	0B	0C	0A	0C	0A	0B	0B	0C	0A	0C	0A	0B	0B	0C	0A	0C	0A	0B
06	0D	0E	0E	0F	0D	0F	0D	0E	0E	0F	0D	0F	0D	0E	0E	10	0D	0F	0D	0F	0E	10	0E	0F	0D	0F
07	10	11	11	13	10	12	10	12	11	13	11	12	10	12	11	13	11	12	10	12	11	13	11	12	10	12
08	13	15	14	16	14	15	13	15	14	16	14	15	13	15	14	16	14	16	13	15	15	16	14	16	14	15
09	16	18	17	19	17	19	16	18	18	19	17	19	17	18	18	19	17	19	17	18	18	19	17	19	17	18
0A	1A	1B	1B	1C	1A	1C	1A	1B	1B	1C	1A	1C	1A	1B	1B	1D	1A	1C	1A	1C	1B	1D	1B	1C	1A	1C
0B	1D	1E	1E	20	1D	1F	1D	1F	1E	20	1E	1F	1D	1F	1E	20	1E	1F	1D	1F	1E	20	1E	1F	1D	1F
0C	20	22	21	23	21	22	20	22	21	23	21	22	20	22	21	23	21	23	20	22	22	23	21	23	21	22
0D	23	25	24	26	24	26	23	25	25	26	24	26	24	25	25	26	24	26	24	25	25	26	24	26	24	25
0E	27	28	28	29	27	29	27	28	28	29	27	29	27	28	28	2A	27	29	27	29	28	2A	28	29	27	29
0F	2A	2B	2B	2D	2A	2C	2A	2C	2B	2D	2B	2C	2A	2C	2B	2D	2B	2C	2A	2C	2B	2D	2B	2C	2A	2C
10	2D	2F	2E	30	2E	2F	2D	2F	2E	30	2E	2F	2D	2F	2E	30	2E	30	2D	2F	2F	30	2E	30	2E	2F
11	30	32	31	33	31	33	30	32	32	33	31	33	31	32	32	33	31	33	31	32	32	33	31	33	31	32
12	34	35	35	36	34	36	34	35	35	36	34	36	34	35	35	37	34	36	34	36	35	37	35	36	34	36
13	37	38	38	3A	37	39	37	39	38	3A	38	39	37	39	38	3A	38	39	37	39	38	3A	38	39	37	39
14	3A	3C	3B	3D	3B	3C	3A	3C	3B	3D	3B	3C	3A	3C	3B	3D	3B	3D	3A	3C	3C	3D	3B	3D	3B	3C
15	3D	3F	3E	40	3E	40	3D	3F	3F	40	3E	40	3E	3F	3F	40	3E	40	3E	3F	3F	40	3E	40	3E	3F
16	41	42	42	43	41	43	41	42	42	43	41	43	41	42	42	44	41	43	41	43	42	44	42	43	41	43
17	44	45	45	47	44	46	44	46	45	47	45	46	44	46	45	47	45	46	44	46	45	47	45	46	44	46
18	47	49	48	4A	48	49	47	49	48	4A	48	49	47	49	48	4A	48	4A	47	49	49	4A	48	4A	48	49
19	4A	4C	4B	4D	4B	4D	4A	4C	4C	4D	4B	4D	4B	4C	4C	4D	4B	4D	4B	4C	4C	4D	4B	4D	4B	4C
1A	4E	4F	4F	50	4E	50	4E	4F	4F	50	4E	50	4E	4F	4F	51	4E	50	4E	50	4F	51	4F	50	4E	50
1B	51	52	52	54	51	53	51	53	52	54	52	53	51	53	52	54	52	53	51	53	52	54	52	53	51	53
1C	54	56	55	57	55	56	54	56	55	57	55	56	54	56	55	57	55	57	54	56	56	57	55	57	55	56
1D	57	59	58	5A	58	5A	57	59	59	5A	58	5A	58	59	59	5A	58	5A	58	59	59	5A	58	5A	58	59
1E	5B	5C	5C	5D	5B	5D	5B	5C	5C	5D	5B	5D	5B	5C	5C	5E	5B	5D	5B	5D	5C	5E	5C	5D	5B	5D
1F	5E	5F	5F	61	5E	60	5E	60	5F	61	5F	60	5E	60	5F	61	5F	60	5E	60	5F	61	5F	60	5E	60
20	61	63	62	64	62	63	61	63	62	64	62	63	61	63	62	64	62	64	61	63	63	64	62	64	62	63
21	64	66	65	67	65	67	64	66	66	67	65	67	65	66	66	67	65	67	65	66	66	67	65	67	65	66
22	68	69	69	6A	68	6A	68	69	69	6A	68	6A	68	69	69	6B	68	6A	68	6A	69	6B	69	6A	68	6A
23	6B	6C	6C	6E	6B	6D	6B	6D	6C	6E	6C	6D	6B	6D	6C	6E	6C	6D	6B	6D	6C	6E	6C	6D	6B	6D
24	6E	70	6F	71	6F	70	6E	70	6F	71	6F	70	6E	70	6F	71	6F	71	6E	70	70	71	6F	71	6F	70
25	71	73	72	74	72	74	71	73	73	74	72	74	72	73	73	74	72	74	72	73	73	74	72	74	72	73
26	75	76	76	77	75	77	75	76	76	77	75	77	75	76	76	78	75	77	75	77	76	78	76	77	75	77
27	78	79	79	7B	78	7A	78	7A	79	7B	79	7A	78	7A	79	7B	79	7A	78	7A	79	7B	79	7A	78	7A
28	7B	7D	7C	7E	7C	7D	7B	7D	7C	7E	7C	7D	7B	7D	7C	7E	7C	7E	7B	7D	7D	7E	7C	7E	7C	7D
29	7E	80	7F	81	7F	81	7E	80	80	81	7F	81	7F	80	80	81	7F	81	7F	80	80	81	7F	81	7F	80
2A	82	83	83	84	82	84	82	83	83	84	82	84	82	83	83	85	82	84	82	84	83	85	83	84	82	84
2B	85	86	86	88	85	87	85	87	86	88	86	87	85	87	86	88	86	87	85	87	86	88	86	87	85	87
2C	88	8A	89	8B	89	8A	88	8A	89	8B	89	8A	88	8A	89	8B	89	8B	88	8A	8A	8B	89	8B	89	8A
2D	8B	8D	8C	8E	8C	8E	8B	8D	8D	8E	8C	8E	8C	8D	8D	8E	8C	8E	8C	8D	8D	8E	8C	8E	8C	8D
2E	8F	90	90	91	8F	91	8F	90	90	91	8F	91	8F	90	90	92	8F	91	8F	91	90	92	90	91	8F	91
2F	92	93	93	95	92	94	92	94	93	95	93	94	92	94	93	95	93	94	92	94	93	95	93	94	92	94
30	95	97	96	98	96	97	95	97	96	98	96	97	95	97	96	98	96	98	95	97	97	98	96	98	96	97
31	98	9A	99	9B	99	9B	98	9A	9A	9B	99	9B	99	9A	9A	9B	99	9B	99	9A	9A	9B	99	9B	99	9A
32	9C	9D	9D	9E	9C	9E	9C	9D	9D	9E	9C	9E	9C	9D	9D	9F	9C	9E	9C	9E	9D	9F	9D	9E	9C	9E
33	9F	A0	A0	A2	9F	A1	9F	A1	A0	A2	A0	A1	9F	A1	A0	A2	A0	A1	9F	A1	A0	A2	A0	A1	9F	A1
34	A2	A4	A3	A5	A3	A4	A2	A4	A3	A5	A3	A4	A2	A4	A3	A5	A3	A5	A2	A4	A4	A5	A3	A5	A3	A4
35	A5	A7	A6	A8	A6	A8	A5	A7	A7	A8	A6	A8	A6	A7	A7	A8	A6	A8	A6	A7	A7	A8	A6	A8	A6	A7
36	A9	AA	AA	AB	A9	AB	A9	AA	AA	AB	A9	AB	A9	AA	AA	AC	A9	AB	A9	AB	AA	AC	AA	AB	A9	AB
37	AC	AD	AD	AF	AC	AE	AC	AE	AD	AF	AD	AE	AC	AE	AD	AF	AD	AE	AC	AE	AD	AF	AD	AE	AC	AE
38	AF	B1	B0	B2	B0	B1	AF	B1	B0	B2	B0	B1	AF	B1	B0	B2	B0	B2	AF	B1	B1	B2	B0	B2	B0	B1
39	B2	B4	B3	B5	B3	B5	B2	B4	B4	B5	B3	B5	B3	B4	B4	B5	B3	B5	B3	B4	B4	B5	B3	B5	B3	B4
3A	B6	B7	B7	B8	B6	B8	B6	B7	B7	B8	B6	B8	B6	B7	B7	B9	B6	B8	B6	B8	B7	B9	B7	B8	B6	B8
3B	B9	BA	BA	BC	B9	BB	B9	BB	BA	BC	BA	BB	B9	BB	BA	BC	BA	BB	B9	BB	BA	BC	BA	BB	B9	BB
3C	BC	BE	BD	BF	BD	BE	BC	BE	BD	BF	BD	BE	BC	BE	BD	BF	BD	BF	BC	BE	BE	BF	BD	BF	BD	BE
3D	BF	C1	C0	C2	C0	C2	BF	C1	C1	C2	C0	C2	C0	C1	C1	C2	C0	C2	C0	C1	C1	C2	C0	C2	C0	C1

ตารางที่ 1 (ต่อ) คลัสเตอร์สำหรับดิสเก็ตต์ขนาด 8 นิ้ว

เซกเตอร์

แทร็ค	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
3E	C3	C4	C4	C5	C3	C5	C3	C4	C4	C5	C3	C5	C3	C4	C4	C6	C3	C5	C3	C5	C4	C6	C4	C5	C3	C5
3F	C6	C7	C7	C9	C6	C8	C6	C8	C7	C9	C7	C8	C6	C8	C7	C9	C7	C8	C6	C8	C7	C9	C7	C8	C6	C8
40	C9	CB	CA	CC	CA	CB	C9	CB	CA	CC	CA	CB	C9	CB	CA	CC	CA	CC	C9	CB	CB	CC	CA	CC	CA	CB
41	CC	CE	CD	CF	CD	CF	CC	CE	CE	CF	CD	CF	CD	CE	CE	CF	CD	CF	CD	CE	CE	CF	CD	CF	CD	CE
42	D0	D1	D1	D2	D0	D2	D0	D1	D1	D2	D0	D2	D0	D1	D1	D3	D0	D2	D0	D2	D1	D3	D1	D2	D0	D2
43	D3	D4	D4	D6	D3	D5	D3	D5	D4	D6	D4	D5	D3	D5	D4	D6	D4	D5	D3	D5	D4	D6	D4	D5	D3	D5
44	D6	D8	D7	D9	D7	D8	D6	D8	D7	D9	D7	D8	D6	D8	D7	D9	D7	D9	D6	D8	D8	D9	D7	D9	D7	D8
45	D9	DB	DA	DC	DA	DC	D9	DB	DB	DC	DA	DC	DA	DB	DB	DC	DA	DC	DA	DB	DB	DC	DA	DC	DA	DB
46	DD	DE	DE	DF	DD	DF	DD	DE	DE	DF	DD	DF	DD	DE	DE	E0	DD	DF	DD	DF	DE	E0	DE	DF	DD	DF
47	E0	E1	E1	E3	E0	E2	E0	E2	E1	E3	E1	E2	E0	E2	E1	E3	E1	[illegible]	E0	E2	E1	E3	E1	E2	E0	E2
48	E3	E5	E4	E6	E4	E5	E3	E5	E4	E6	E4	E5	E3	E5	E4	E6	E4	E6	E3	E5	E5	E6	E4	E6	E4	E5
49	E6	E8	E7	E9	E7	E9	E6	E8	E8	E9	E7	E9	E7	E8	E8	E9	E7	E9	E7	E8	E8	E9	E7	E9	E7	E8
4A	EA	EB	EB	EC	EA	EC	EA	EB	EB	EC	EA	EC	EA	EB	EB	ED	EA	EC	EA	EC	EB	ED	EB	EC	EA	EC
4B	ED	EE	EE	F0	ED	EF	ED	EF	EE	F0	EE	EF	ED	EF	EE	F0	EE	EF	ED	EF	EE	F0	EE	EF	ED	EF
4C	F0	F2	F1	--	F1	F2	F0	F2	F1	--	F1	F2	F0	F2	F1	--	F1	--	F0	F2	F2	--	F1	--	F1	F2

ตารางที่ 2 ตารางการจัดคลัสเตอร์ของดิสเก็ตต์ขนาด 8 นิ้ว

เซกเตอร์

คลัสเตอร์	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc
00	02,01	02,07	02,0D	02,13	02,19	02,05	02,0B	02,11
01	02,17	02,03	02,09	02,0F	02,15	02,02	02,08	02,0E
02	02,14	02,1A	02,06	02,0C	02,12	02,18	02,04	02,0A
03	02,10	02,16	03,01	03,07	03,0D	03,13	03,19	03,05
04	03,0B	03,11	03,17	03,03	03,09	03,0F	03,15	03,02
05	03,08	03,0E	03,14	03,1A	03,06	03,0C	03,12	03,18
06	03,04	03,0A	03,10	03,16	04,01	04,07	04,0D	04,13
07	04,19	04,05	04,0B	04,11	04,17	04,03	04,09	04,0F
08	04,15	04,02	04,08	04,0E	04,14	04,1A	04,06	04,0C
09	04,12	04,18	04,04	04,0A	04,10	04,16	05,01	05,07
0A	05,0D	05,13	05,19	05,05	05,0B	05,11	05,17	05,03
0B	05,09	05,0F	05,15	05,02	05,08	05,0E	05,14	05,1A
0C	05,06	05,0C	05,12	05,18	05,04	05,0A	05,10	05,16
0D	06,01	06,07	06,0D	06,13	06,19	06,05	06,0B	06,11
0E	06,17	06,03	06,09	06,0F	06,15	06,02	06,08	06,0E
0F	06,14	06,1A	06,06	06,0C	06,12	06,18	06,04	06,0A
10	06,10	06,16	07,01	07,07	07,0D	07,13	07,19	07,05
11	07,0B	07,11	07,17	07,03	07,09	07,0F	07,15	07,02
12	07,08	07,0E	07,14	07,1A	07,06	07,0C	07,12	07,18
13	07,04	07,0A	07,10	07,16	08,01	08,07	08,0D	08,13
14	08,19	08,05	08,0B	08,11	08,17	08,03	08,09	08,0F
15	08,15	08,02	08,08	08,0E	08,14	08,1A	08,06	08,0C
16	08,12	08,18	08,04	08,0A	08,10	08,16	09,01	09,07
17	09,0D	09,13	09,19	09,05	09,0B	09,11	09,17	09,03
18	09,09	09,0F	09,15	09,02	09,08	09,0E	09,14	09,1A
19	09,06	09,0C	09,12	09,18	09,04	09,0A	09,10	09,16
1A	0A,01	0A,07	0A,0D	0A,13	0A,19	0A,05	0A,0B	0A,11
1B	0A,17	0A,03	0A,09	0A,0F	0A,15	0A,02	0A,08	0A,0E
1C	0A,14	0A,1A	0A,06	0A,0C	0A,12	0A,18	0A,04	0A,0A
1D	0A,10	0A,16	0B,01	0B,07	0B,0D	0B,13	0B,19	0B,05
1E	0B,0B	0B,11	0B,17	0B,03	0B,09	0B,0F	0B,15	0B,02

ตารางที่ 2 (ต่อ) ตารางการจัดคลัสเตอร์ของดิสเก็ตต์ขนาด 8 นิ้ว

คลัสเตอร์	เซกเตอร์							
	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc
1F	0B,08	0B,0E	0B,14	0B,1A	0B,06	0B,0C	0B,12	0B,18
20	0B,04	0B,0A	0B,10	0B,16	0C,01	0C,07	0C,0D	0C,13
21	0C,19	0C,05	0C,0B	0C,11	0C,17	0C,03	0C,09	0C,0F
22	0C,15	0C,02	0C,08	0C,0E	0C,14	0C,1A	0C,06	0C,0C
23	0C,12	0C,18	0C,04	0C,0A	0C,10	0C,16	0D,01	0D,07
24	0D,0D	0D,13	0D,19	0D,05	0D,0B	0D,11	0D,17	0D,03
25	0D,09	0D,0F	0D,15	0D,02	0D,08	0D,0E	0D,14	0D,1A
26	0D,06	0D,0C	0D,12	0D,18	0D,04	0D,0A	0D,10	0D,16
27	0E,01	0E,07	0E,0D	0E,13	0E,19	0E,05	0E,0B	0E,11
28	0E,17	0E,03	0E,09	0E,0F	0E,15	0E,02	0E,08	0E,0E
29	0E,14	0E,1A	0E,06	0E,0C	0E,12	0E,18	0E,04	0E,0A
2A	0E,10	0E,16	0F,01	0F,07	0F,0D	0F,13	0F,19	0F,05
2B	0F,0B	0F,11	0F,17	0F,03	0F,09	0F,0F	0F,15	0F,02
2C	0F,08	0F,0E	0F,14	0F,1A	0F,06	0F,0C	0F,12	0F,18
2D	0F,04	0F,0A	0F,10	0F,16	10,01	10,07	10,0D	10,13
2E	10,19	10,05	10,0B	10,11	10,17	10,03	10,09	10,0F
2F	10,15	10,02	10,08	10,0E	10,14	10,1A	10,06	10,0C
30	10,12	10,18	10,04	10,0A	10,10	10,16	11,01	11,07
31	11,0D	11,13	11,19	11,05	11,0B	11,11	11,17	11,03
32	11,09	11,0F	11,15	11,02	11,08	11,0E	11,14	11,1A
33	11,06	11,0C	11,12	11,18	11,04	11,0A	11,10	11,16
34	12,01	12,07	12,0D	12,13	12,19	12,05	12,0B	12,11
35	12,17	12,03	12,09	12,0F	12,15	12,02	12,08	12,0E
36	12,14	12,1A	12,06	12,0C	12,12	12,18	12,04	12,0A
37	12,10	12,16	13,01	13,07	13,0D	13,13	13,19	13,05
38	13,0B	13,11	13,17	13,03	13,09	13,0F	13,15	13,02
39	13,08	13,0E	13,14	13,1A	13,06	13,0C	13,12	13,18
3A	13,04	13,0A	13,10	13,16	14,01	14,07	14,0D	14,13
3B	14,19	14,05	14,0B	14,11	14,17	14,03	14,09	14,0F
3C	14,15	14,02	14,08	14,0E	14,14	14,1A	14,06	14,0C
3D	14,12	14,18	14,04	14,0A	14,10	14,16	15,01	15,07
3E	15,0D	15,13	15,19	15,05	15,0B	15,11	15,17	15,03
3F	15,09	15,0F	15,15	15,02	15,08	15,0E	15,14	15,1A
40	15,06	15,0C	15,12	15,18	15,04	15,0A	15,10	15,16
41	16,01	16,07	16,0D	16,13	16,19	16,05	16,0B	16,11
42	16,17	16,03	16,09	16,0F	16,15	16,02	16,08	16,0E
43	16,14	16,1A	16,06	16,0C	16,12	16,18	16,04	16,0A
44	16,10	16,16	17,01	17,07	17,0D	17,13	17,19	17,05
45	17,0B	17,11	17,17	17,03	17,09	17,0F	17,15	17,02
46	17,08	17,0E	17,14	17,1A	17,06	17,0C	17,12	17,18
47	17,04	17,0A	17,10	17,16	18,01	18,07	18,0D	18,13
48	18,19	18,05	18,0B	18,11	18,17	18,03	18,09	18,0F
49	18,15	18,02	18,08	18,0E	18,14	18,1A	18,06	18,0C
4A	18,12	18,18	18,04	18,0A	18,10	18,16	19,01	19,07
4B	19,0D	19,13	19,19	19,05	19,0B	19,11	19,17	19,03
4C	19,09	19,0F	19,15	19,02	19,08	19,0E	19,14	19,1A
4D	19,06	19,0C	19,12	19,18	19,04	19,0A	19,10	19,16

ตารางที่ 2 (ต่อ) ตารางการจัดคลัสเตอร์ของดิสเก็ตต์ขนาด 8 นิ้ว

เซกเตอร์

คลัสเตอร์	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc
4E	1A,01	1A,07	1A,0D	1A,13	1A,19	1A,05	1A,0B	1A,11
4F	1A,17	1A,03	1A,09	1A,0F	1A,15	1A,02	1A,08	1A,0E
50	1A,14	1A,1A	1A,06	1A,0C	1A,12	1A,18	1A,04	1A,0A
51	1A,10	1A,16	1B,01	1B,07	1B,0D	1B,13	1B,19	1B,05
52	1B,0B	1B,11	1B,17	1B,03	1B,09	1B,0F	1B,15	1B,02
53	1B,08	1B,0E	1B,14	1B,1A	1B,06	1B,0C	1B,12	1B,18
54	1B,04	1B,0A	1B,10	1B,16	1C,01	1C,07	1C,0D	1C,13
55	1C,19	1C,05	1C,0B	1C,11	1C,17	1C,03	1C,09	1C,0F
56	1C,15	1C,02	1C,08	1C,0E	1C,14	1C,1A	1C,06	1C,0C
57	1C,12	1C,18	1C,04	1C,0A	1C,10	1C,16	1D,01	1D,07
58	1D,0D	1D,13	1D,19	1D,05	1D,0B	1D,11	1D,17	1D,03
59	1D,09	1D,0F	1D,15	1D,02	1D,08	1D,0E	1D,14	1D,1A
5A	1D,06	1D,0C	1D,12	1D,18	1D,04	1D,0A	1D,10	1D,16
5B	1E,01	1E,07	1E,0D	1E,13	1E,19	1E,05	1E,0B	1E,11
5C	1E,17	1E,03	1E,09	1E,0F	1E,15	1E,02	1E,08	1E,0E
5D	1E,14	1E,1A	1E,06	1E,0C	1E,12	1E,18	1E,04	1E,0A
5E	1E,10	1E,16	1F,01	1F,07	1F,0D	1F,13	1F,19	1F,05
5F	1F,0B	1F,11	1F,17	1F,03	1F,09	1F,0F	1F,15	1F,02
60	1F,08	1F,0E	1F,14	1F,1A	1F,06	1F,0C	1F,12	1F,18
61	1F,04	1F,0A	1F,10	1F,16	20,01	20,07	20,0D	20,13
62	20,19	20,05	20,0B	20,11	20,17	20,03	20,09	20,0F
63	20,15	20,02	20,08	20,0E	20,14	20,1A	20,06	20,0C
64	20,12	20,18	20,04	20,0A	20,10	20,16	21,01	21,07
65	21,0D	21,13	21,19	21,05	21,0B	21,11	21,17	21,03
66	21,09	21,0F	21,15	21,02	21,08	21,0E	21,14	21,1A
67	21,06	21,0C	21,12	21,18	21,04	21,0A	21,10	21,16
68	22,01	22,07	22,0D	22,13	22,19	22,05	22,0B	22,11
69	22,17	22,03	22,09	22,0F	22,15	22,02	22,08	22,0E
6A	22,14	22,1A	22,06	22,0C	22,12	22,18	22,04	22,0A
6B	22,10	22,16	23,01	23,07	23,0D	23,13	23,19	23,05
6C	23,0B	23,11	23,17	23,03	23,09	23,0F	23,15	23,02
6D	23,08	23,0E	23,14	23,1A	23,06	23,0C	23,12	23,18
6E	23,04	23,0A	23,10	23,16	24,01	24,07	24,0D	24,13
6F	24,19	24,05	24,0B	24,11	24,17	24,03	24,09	24,0F
70	24,15	24,02	24,08	24,0E	24,14	24,1A	24,06	24,0C
71	24,12	24,18	24,04	24,0A	24,10	24,16	25,01	25,07
72	25,0D	25,13	25,19	25,05	25,0B	25,11	25,17	25,03
73	25,09	25,0F	25,15	25,02	25,08	25,0E	25,14	25,1A
74	25,06	25,0C	25,12	25,18	25,04	25,0A	25,10	25,16
75	26,01	26,07	26,0D	26,13	26,19	26,05	26,0B	26,11
76	26,17	26,03	26,09	26,0F	26,15	26,02	26,08	26,0E
77	26,14	26,1A	26,06	26,0C	26,12	26,18	26,04	26,0A
78	26,10	26,16	27,01	27,07	27,0D	27,13	27,19	27,05
79	27,0B	27,11	27,17	27,03	27,09	27,0F	27,15	27,02
7A	27,08	27,0E	27,14	27,1A	27,06	27,0C	27,12	27,18
7B	27,04	27,0A	27,10	27,16	28,01	28,07	28,0D	28,13
7C	28,19	28,05	28,0B	28,11	28,17	28,03	28,09	28,0F

ตารางที่ 2 (ต่อ) ตารางการจัดคลัสเตอร์ของดิสเก็ตต์ขนาด 8 นิ้ว

คลัสเตอร์	เซกเตอร์							
	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc
7D	28,15	28,02	28,08	28,0E	28,14	28,1A	28,06	28,0C
7E	28,12	28,18	28,04	28,0A	28,10	28,16	29,01	29,07
7F	29,0D	29,13	29,19	29,05	29,0B	29,11	29,17	29,03
80	29,09	29,0F	29,15	29,02	29,08	29,0E	29,14	29,1A
81	29,06	29,0C	29,12	29,18	29,04	29,0A	29,10	29,16
82	2A,01	2A,07	2A,0D	2A,13	2A,19	2A,05	2A,0B	2A,11
83	2A,17	2A,03	2A,09	2A,0F	2A,15	2A,02	2A,08	2A,0E
84	2A,14	2A,1A	2A,06	2A,0C	2A,12	2A,18	2A,04	2A,0A
85	2A,10	2A,16	2B,01	2B,07	2B,0D	2B,13	2B,19	2B,05
86	2B,0B	2B,11	2B,17	2B,03	2B,09	2B,0F	2B,15	2B,02
87	2B,08	2B,0E	2B,14	2B,1A	2B,06	2B,0C	2B,12	2B,18
88	2B,04	2B,0A	2B,10	2B,16	2C,01	2C,07	2C,0D	2C,13
89	2C,19	2C,05	2C,0B	2C,11	2C,17	2C,03	2C,09	2C,0F
8A	2C,15	2C,02	2C,08	2C,0E	2C,14	2C,1A	2C,06	2C,0C
8B	2C,12	2C,18	2C,04	2C,0A	2C,10	2C,16	2D,01	2D,07
8C	2D,0D	2D,13	2D,19	2D,05	2D,0B	2D,11	2D,17	2D,03
8D	2D,09	2D,0F	2D,15	2D,02	2D,08	2D,0E	2D,14	2D,1A
8E	2D,06	2D,0C	2D,12	2D,18	2D,04	2D,0A	2D,10	2D,16
8F	2E,01	2E,07	2E,0D	2E,13	2E,19	2E,05	2E,0B	2E,11
90	2E,17	2E,03	2E,09	2E,0F	2E,15	2E,02	2E,08	2E,0E
91	2E,14	2E,1A	2E,06	2E,0C	2E,12	2E,18	2E,04	2E,0A
92	2E,10	2E,16	2F,01	2F,07	2F,0D	2F,13	2F,19	2F,05
93	2F,0B	2F,11	2F,17	2F,03	2F,09	2F,0F	2F,15	2F,02
94	2F,08	2F,0E	2F,14	2F,1A	2F,06	2F,0C	2F,12	2F,18
95	2F,04	2F,0A	2F,10	2F,16	30,01	30,07	30,0D	30,13
96	30,19	30,05	30,0B	30,11	30,17	30,03	30,09	30,0F
97	30,15	30,02	30,08	30,0E	30,14	30,1A	30,06	30,0C
98	30,12	30,18	30,04	30,0A	30,10	30,16	31,01	31,07
99	31,0D	31,13	31,19	31,05	31,0B	31,11	31,17	31,03
9A	31,09	31,0F	31,15	31,02	31,08	31,0E	31,14	31,1A
9B	31,06	31,0C	31,12	31,18	31,04	31,0A	31,10	31,16
9C	32,01	32,07	32,0D	32,13	32,19	32,05	32,0B	32,11
9D	32,17	32,03	32,09	32,0F	32,15	32,02	32,08	32,0E
9E	32,14	32,1A	32,06	32,0C	32,12	32,18	32,04	32,0A
9F	32,10	32,16	33,01	33,07	33,0D	33,13	33,19	33,05
A0	33,0B	33,11	33,17	33,03	33,09	33,0F	33,15	33,02
A1	33,08	33,0E	33,14	33,1A	33,06	33,0C	33,12	33,18
A2	33,04	33,0A	33,10	33,16	34,01	34,07	34,0D	34,13
A3	34,19	34,05	34,0B	34,11	34,17	34,03	34,09	34,0F
A4	34,15	34,02	34,08	34,0E	34,14	34,1A	34,06	34,0C
A5	34,12	34,18	34,04	34,0A	34,10	34,16	35,01	35,07
A6	35,0D	35,13	35,19	35,05	35,0B	35,11	35,17	35,03
A7	35,09	35,0F	35,15	35,02	35,08	35,0E	35,14	35,1A
A8	35,06	35,0C	35,12	35,18	35,04	35,0A	35,10	35,16
A9	36,01	36,07	36,0D	36,13	36,19	36,05	36,0B	36,11
AA	36,17	36,03	36,09	36,0F	36,15	36,02	36,08	36,0E

ตารางที่ 2 (ต่อ) ตารางการจัดคลัสเตอร์ของดิสเก็ตต์ขนาด 8 นิ้ว

	เซกเตอร์							
คลัสเตอร์	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc
AB	36,14	36,1A	36,06	36,0C	36,12	36,18	36,04	36,0A
AC	36,10	36,16	37,01	37,07	37,0D	37,13	37,19	37,05
AD	37,0B	37,11	37,17	37,03	37,09	37,0F	37,15	37,02
AE	37,08	37,0E	37,14	37,1A	37,06	37,0C	37,12	37,18
AF	37,04	37,0A	37,10	37,16	38,01	38,07	38,0D	38,13
B0	38,19	38,05	38,0B	38,11	38,17	38,03	38,09	38,0F
B1	38,15	38,02	38,08	38,0E	38,14	38,1A	38,06	38,0C
B2	38,12	38,18	38,04	38,0A	38,10	38,16	39,01	39,07
B3	39,0D	39,13	39,19	39,05	39,0B	39,11	39,17	39,03
B4	39,09	39,0F	39,15	39,02	39,08	39,0E	39,14	39,1A
B5	39,06	39,0C	39,12	39,18	39,04	39,0A	39,10	39,16
B6	3A,01	3A,07	3A,0D	3A,13	3A,19	3A,05	3A,0B	3A,11
B7	3A,17	3A,03	3A,09	3A,0F	3A,15	3A,02	3A,08	3A,0E
B8	3A,14	3A,1A	3A,06	3A,0C	3A,12	3A,18	3A,04	3A,0A
B9	3A,10	3A,16	3B,01	3B,07	3B,0D	3B,13	3B,19	3B,05
BA	3B,0B	3B,11	3B,17	3B,03	3B,09	3B,0F	3B,15	3B,02
BB	3B,08	3B,0E	3B,14	3B,1A	3B,06	3B,0C	3B,12	3B,18
BC	3B,04	3B,0A	3B,10	3B,16	3C,01	3C,07	3C,0D	3C,13
BD	3C,19	3C,05	3C,0B	3C,11	3C,17	3C,03	3C,09	3C,0F
BE	3C,15	3C,02	3C,08	3C,0E	3C,14	3C,1A	3C,06	3C,0C
BF	3C,12	3C,18	3C,04	3C,0A	3C,10	3C,16	3D,01	3D,07
C0	3D,0D	3D,13	3D,19	3D,05	3D,0B	3D,11	3D,17	3D,03
C1	3D,09	3D,0F	3D,15	3D,02	3D,08	3D,0E	3D,14	3D,1A
C2	3D,06	3D,0C	3D,12	3D,18	3D,04	3D,0A	3D,10	3D,16
C3	3E,01	3E,07	3E,0D	3E,13	3E,19	3E,05	3E,0B	3E,11
C4	3E,17	3E,03	3E,09	3E,0F	3E,15	3E,02	3E,08	3E,0E
C5	3E,14	3E,1A	3E,06	3E,0C	3E,12	3E,18	3E,04	3E,0A
C6	3E,10	3E,16	3F,01	3F,07	3F,0D	3F,13	3F,19	3F,05
C7	3F,0B	3F,11	3F,17	3F,03	3F,09	3F,0F	3F,15	3F,02
C8	3F,08	3F,0E	3F,14	3F,1A	3F,06	3F,0C	3F,12	3F,18
C9	3F,04	3F,0A	3F,10	3F,16	40,01	40,07	40,0D	40,13
CA	40,19	40,05	40,0B	40,11	40,17	40,03	40,09	40,0F
CB	40,15	40,02	40,08	40,0E	40,14	40,1A	40,06	40,0C
CC	40,12	40,18	40,04	40,0A	40,10	40,16	41,01	41,07
CD	41,0D	41,13	41,19	41,05	41,0B	41,11	41,17	41,03
CE	41,09	41,0F	41,15	41,02	41,08	41,0E	41,14	41,1A
CF	41,06	41,0C	41,12	41,18	41,04	41,0A	41,10	41,16
D0	42,01	42,07	42,0D	42,13	42,19	42,05	42,0B	42,11
D1	42,17	42,03	42,09	42,0F	42,15	42,02	42,08	42,0E
D2	42,14	42,1A	42,06	42,0C	42,12	42,18	42,04	42,0A
D3	42,10	42,16	43,01	43,07	43,0D	43,13	43,19	43,05
D4	43,0B	43,11	43,17	43,03	43,09	43,0F	43,15	43,02
D5	43,08	43,0E	43,14	43,1A	43,06	43,0C	43,12	43,18
D6	43,04	43,0A	43,10	43,16	44,01	44,07	44,0D	44,13
D7	44,19	44,05	44,0B	44,11	44,17	44,03	44,09	44,0F
D8	44,15	44,02	44,08	44,0E	44,14	44,1A	44,06	44,0C

ตารางที่ 2 (ต่อ) ตารางการจัดคลัสเตอร์ของดิสเก็ตต์ขนาด 8 นิ้ว

	เซกเตอร์							
คลัสเตอร์	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc	Tk,Sc
D9	44,12	44,18	44,04	44,0A	44,10	44,16	45,01	45,07
DA	45,0D	45,13	45,19	45,05	45,0B	45,11	45,17	45,03
DB	45,09	45,0F	45,15	45,02	45,08	45,0E	45,14	45,1A
DC	45,06	45,0C	45,12	45,18	45,04	45,0A	45,10	45,16
DD	46,01	46,07	46,0D	46,13	46,19	46,05	46,0B	46,11
DE	46,17	46,03	46,09	46,0F	46,15	46,02	46,08	46,0E
DF	46,14	46,1A	46,06	46,0C	46,12	46,18	46,04	46,0A
E0	46,10	46,16	47,01	47,07	47,0D	47,13	47,19	47,05
E1	47,0B	47,11	47,17	47,03	47,09	47,0F	47,15	47,02
E2	47,08	47,0E	47,14	47,1A	47,06	47,0C	47,12	47,18
E3	47,04	47,0A	47,10	47,16	48,01	48,07	48,0D	48,13
E4	48,19	48,05	48,0B	48,11	48,17	48,03	48,09	48,0F
E5	48,15	48,02	48,08	48,0E	48,14	48,1A	48,06	48,0C
E6	48,12	48,18	48,04	48,0A	48,10	48,16	49,01	49,07
E7	49,0D	49,13	49,19	49,05	49,0B	49,11	49,17	49,03
E8	49,09	49,0F	49,15	49,02	49,08	49,0E	49,14	49,1A
E9	49,06	49,0C	49,12	49,18	49,04	49,0A	49,10	49,16
EA	4A,01	4A,07	4A,0D	4A,13	4A,19	4A,05	4A,0B	4A,11
EB	4A,17	4A,03	4A,09	4A,0F	4A,15	4A,02	4A,08	4A,0E
EC	4A,14	4A,1A	4A,06	4A,0C	4A,12	4A,18	4A,04	4A,0A
ED	4A,10	4A,16	4B,01	4B,07	4B,0D	4B,13	4B,19	4B,05
EE	4B,0B	4B,11	4B,17	4B,03	4B,09	4B,0F	4B,15	4B,02
EF	4B,08	4B,0E	4B,14	4B,1A	4B,06	4B,0C	4B,12	4B,18
F0	4B,04	4B,0A	4B,10	4B,16	4C,01	4C,07	4C,0D	4C,13
F1	4C,19	4C,05	4C,0B	4C,11	4C,17	4C,03	4C,09	4C,0F
F2	4C,15	4C,02	4C,08	4C,0E	4C,14	4C,1A	4C,06	4C,0C

หนังสือสำหรับผู้ศึกษา/ใช้/เล่นคอมพิวเตอร์



โปรแกรมคอมพิวเตอร์ ภาษาเบสิค

โดย ยืน ภู่วรวรรณ พิชิต สุขเจริญพงศ์ สมนึก คีรีโต และภิญโญ ศรีสุขินวงศ์ ราคา 55 บาท

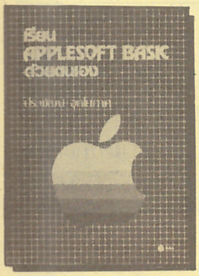
เป็นหนังสือภาษาเบสิคอีกเล่มหนึ่งที่จัดทำขึ้น โดยคณะอาจารย์จากห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตร โดยพยายามสร้างพื้นฐานการเขียนโปรแกรมสำหรับผู้เริ่มต้น การประยุกต์ใช้งานของภาษาเบสิคในเรื่องการเขียนภาพ การสร้างเกม การใช้งานทางคณิตศาสตร์และสถิติ การใช้งานทางด้านธุรกิจ การบริหาร



คอมพิวเตอร์กับการเขียนโปรแกรมภาษาเบสิค

โดย พรทิพย์ เลาหวิโรจน์ และสุพจน์ จิตต์ประเสริฐ ราคา 65 บาท

หนังสือเล่มนี้สำหรับผู้ที่ต้องการเรียนและเขียนภาษา APPLESOFT BASIC ด้วยตนเองให้เป็นภายใน 50 ชั่วโมง โดยจะเริ่มต้นตั้งแต่พื้นฐานของการโปรแกรมด้วยภาษาเบสิค อ้างอิงถึงการใช้งานกับเครื่องยี่ห้อ APPLE โดยจะค่อย ๆ พัฒนาคุณขึ้นมาจนถึงระดับที่จะโปรแกรมงานบางอย่างได้เอง



เรียน APPLESOFT BASIC ด้วยตนเอง (เล่ม 1)

โดย พ.ต.ประพัฒน์ อุทโยภาศ ราคา 45 บาท

หนังสือเล่มนี้จะช่วยให้ท่านสามารถเขียนโปรแกรมสั่งงานไมโครคอมพิวเตอร์ได้ด้วยตัวท่านเอง โดยจะเริ่มตั้งแต่คอมพิวเตอร์คืออะไร เป็นอย่างไร มีอะไรบ้าง จะต้องสั่งงานอย่างไร คำสั่งที่ใช้มีอะไรบ้าง ตลอดเล่มจะมีโปรแกรมที่ทำการทดสอบแล้วโดยผ่านเครื่อง APPLE II ให้ท่านฝึกฝนโดยเริ่มจากง่าย ๆ ขึ้นไป จนในที่สุดท่านก็จะสามารถอ่านโปรแกรมเข้าใจได้ด้วยตัวท่านเอง



เรียน APPLESOFT BASIC ด้วยตนเอง (เล่ม 2)

โดย พ.ต.ประพัฒน์ อุทโยภาศ ราคา 42 บาท

เป็นหนังสือเพิ่มความรู้สำหรับผู้ที่อ่านเล่ม 1 มาแล้ว เพื่อจะได้ศึกษาส่วนที่ผู้เขียนเว้นไว้เพื่อมิให้สับสนในระยะที่เริ่มเรียนรู้ บทต่าง ๆ ภายในเล่มมี ลึกเข้าไปในคอมพิวเตอร์, เพิ่มเติมเสริมแต่ง, FUNCTIONS, กราฟิก, SHAPE TABLE, TEXT FILE และ FLOW CHART

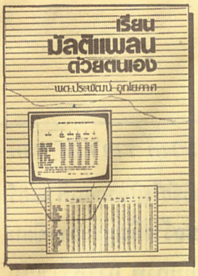


เรียน/เล่น/ใช้ โปรแกรม APPLESOFT BASIC

โดย พ.ต.ประพัฒน์ อุทโยภาศ ราคา 45 บาท

เมื่อท่านสามารถอ่านโปรแกรมเข้าใจแล้ว หนังสือเล่มนี้จะเป็นเล่มต่อไปที่ช่วยเพิ่มประสบการณ์แก่ท่านในด้านการหัดป้อนข้อมูล (KEY) หัด SAVE, LOAD, LIST และ RUN หัดแก้และหัดแปลงโปรแกรม ซึ่งต่อไปก็จะสามารถเขียนโปรแกรมได้เอง สำหรับโปรแกรม 40 โปรแกรมที่ได้รวบรวมไว้ในเล่ม มีทั้งที่เป็นเกม โปรแกรมทางการศึกษา คณิตศาสตร์ สถิติ และธุรกิจ เพราะฉะนั้นท่านสามารถจะเลือกฝึกได้อย่างจุใจ

เรียนมัลติแพลนด้วยตนเอง



โดย ประพัฒน์ อุทโยภาศ ราคา 46 บาท

หากท่านมีนิ้วมือเหลืออย่างน้อย 2 นิ้ว พออ่านภาษาอังกฤษออก มีไมโครคอมพิวเตอร์ที่มีหน่วยความจำอย่างน้อย 64 K BYTE มีโปรแกรมสำเร็จรูปชื่อมัลติแพลน และได้อ่านหนังสือเล่มนี้เพียงครึ่งเล่ม ท่านก็สามารถนำคอมพิวเตอร์ไปใช้ประโยชน์ได้อย่างกว้างขวาง อาทิเช่น ทำบัญชี สินค้าคงคลัง ออกอินวอยซ์ เก็บหมายเลขโทรศัพท์ ฯลฯ หนังสือเล่มนี้ใช้ภาษาง่าย ๆ อย่างเป็นกันเองเหมือนเพื่อนคุยกัน แต่ขณะเดียวกันจะให้ความรู้ที่แม้แต่ในหนังสือคู่มือการใช้โปรแกรมนี้เองก็ไม่ได้กล่าวไว้ กล่าวได้ว่าหนังสือเล่มนี้จะเป็นประโยชน์แก่ทุกคน ตั้งแต่ผู้เริ่มใช้ จนกระทั่งผู้ที่ได้เคยใช้โปรแกรมนี้มาอย่างชัดเจน

รวมโปรแกรมสำหรับแอปเปิ้ล เล่ม 1



โดย เฮาเอิร์ด เบเรนบอน ราคา 48 บาท

และนี่คือ โปรแกรมใช้งานสำหรับนักเล่นไมโครคอมพิวเตอร์ที่ต้องการจะใช้เครื่อง APPLE II ให้คุ้มค่า มี 28 โปรแกรมอยู่ภายในเล่มแบ่งเป็น 6 ภาคตามลักษณะการใช้งาน เช่น ภาคเกี่ยวกับเวลา ธุรกิจและการลงทุน การศึกษา เป็นต้น ตัวอย่างโปรแกรม ได้แก่เครื่องควบคุมการปิดเปิดไฟฟ้าสารพัดประโยชน์ เครื่องตั้งเวลาอัตโนมัติ และคิดค่าเสื่อมราคาวิเคราะห์การใช้ไฟฟ้า หมอดูไพ่ป๊อก และเกมฝึกสมองอีกมากมาย

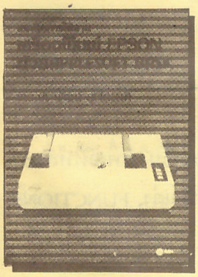
รวมโปรแกรมสำหรับแอปเปิ้ล เล่ม 2



โดย เฮาเอิร์ด เบเรนบอน ราคา 55 บาท

เป็นหนังสือรวมโปรแกรมใช้งานเครื่อง APPLE II อีก 37 โปรแกรมอาทิเช่น การคิดดอกเบี้ยทบต้น โปรแกรมเก็บหมายเลขโทรศัพท์ โปรแกรมรักษาสภาพสมดุลของบัญชีเช็ค โปรแกรมแนะนำการซื้อหุ้น ทดสอบความสามารถทางจิต โปรแกรมแบ่งเกรดนักเรียนและโปรแกรมทดสอบความสามารถอีกมากมาย

เทคนิคการใช้งานเครื่องพิมพ์ EPSON ด้วย APPLESOFT BASIC



โดย พ.ต.ประพัฒน์ อุทโยภาศ ราคา 22 บาท

หนังสือเล่มนี้จะบอกวิธีใช้เครื่องพิมพ์ EPSON ที่พ่วงกับเครื่องคอมพิวเตอร์ตระกูล APPLE มีการสร้างตัวหนังสือแบบต่าง ๆ ถึง 11 แบบ การปรับระยะบรรทัด การพิมพ์ภาพกราฟิกพร้อมกับมีโปรแกรมแถมให้ การพิมพ์นามบัตรสำเร็จรูป การพิมพ์กราฟแท่ง มีตารางสรุปคำสั่งต่าง ๆอย่างที่จะหาไม่ได้ในหนังสือทั้งไทยและเทศ และท้ายเล่มมีโปรแกรมสั้น ๆ สนุก ๆ สำหรับเด็ก ๆ หนังสือเล่มนี้เหมาะสำหรับผู้ที่มีเครื่องพิมพ์ EPSON อยู่แล้วหรือผู้ที่กำลังคิดจะมี

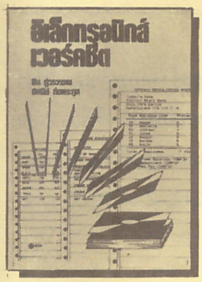
ระบบจัดการฐานข้อมูล dBASE II



โดย ยืน ภู่วรวรรณ และสมนึก คีรีโต ราคา 60 บาท

เป็นหนังสือสำหรับผู้ที่ยังไม่เคยใช้ไมโครคอมพิวเตอร์ในระบบจัดการฐานข้อมูลมาก่อนเลย และสนใจการใช้งาน dBASE II ซึ่งเป็นโปรแกรมที่มีประสิทธิภาพสูงสุดและได้รับความนิยมมากโปรแกรมหนึ่ง ภายในเล่มแบ่งเป็น 3 ตอน ตอนแรกจะวางพื้นฐานของระบบจัดการฐานข้อมูลตลอดจนโปรแกรมโครงสร้าง ตอนที่สองว่าด้วยวิธีการทำงานของ dBASE II และตอนที่สามว่าด้วยการนำเอา dBASE II มาใช้งาน

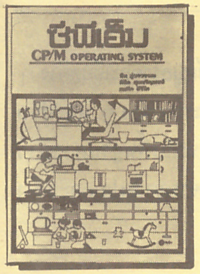
อิเล็กทรอนิกส์เวอร์คชีต



โดย ยืน ภู่วรวรรณ และอัศนีย์ ก่อตระกูล ราคา 59 บาท

เหมาะสำหรับผู้ที่เรียนรู้การใช้งานคอมพิวเตอร์มาบ้างแล้ว และกำลังต้องการนำไมโครคอมพิวเตอร์มาใช้งานจริง โดยจะให้หลักการและวิธีใช้อิเล็กทรอนิกส์เวอร์คชีต ซึ่งใช้ได้ทั้งวิสิแคลค์ มัลติแพลน ซูเปอร์แคลค์ (ทั้งกำลัง 2 และกำลัง 3) และโลตัส 123 ตัวอย่างของอิเล็กทรอนิกส์เวอร์คชีตในหนังสือเล่มนี้ใช้กับโปรแกรมซูเปอร์แคลค์เป็นหลัก แต่ก็มีหลักการใช้หรือการเรียกคล้ายอิเล็กทรอนิกส์เวอร์คชีตอีกหลายตัว

ซีพีเอ็ม (CP/M Operating System)



โดย ยืน ภู่วรวรรณ พิชิต สุขเจริญพงษ์ และสมนึก คีรีโต ราคา 58 บาท

ภายในหนังสือเล่มนี้ได้ให้รายละเอียดเกี่ยวกับโครงสร้างภายในของซีพีเอ็ม การเชื่อมต่อทางซอฟต์แวร์เพื่อพัฒนาโปรแกรมหรือดัดแปลงแก้ไขโปรแกรมการจัดการระบบข้อมูลบนแผ่นดิสเก็ตต์ รายละเอียดเกี่ยวกับคำสั่งต่าง ๆ ที่ใช้ในซีพีเอ็ม ทั้งเป็นคำสั่งที่อยู่ภายในตัวซีพีเอ็มเองและคำสั่งประกอบที่อยู่ภายนอก ทำให้ผู้ศึกษาสามารถเรียนรู้และเข้าใจเกี่ยวกับการพัฒนา หรือเขียนโปรแกรมภาษาแอสเซมบลี้ได้เป็นอย่างดี

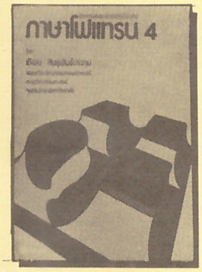
โปรแกรมคอมพิวเตอร์ภาษาฟอร์แทรน



โดย ยืน ภู่วรวรรณ พิชิต สุขเจริญวงศ์ และนนทวัฒน์ จันทร์เจริญ ราคา 68 บาท

หนังสือเล่มนี้เหมาะสำหรับใช้เป็นตำราในระดับวิทยาลัย มหาวิทยาลัย ตลอดจนผู้ที่จะศึกษาด้วยตนเอง เนื้อหาภายในได้วางโครงร่างในลักษณะต่อเรียงกันไปโดยเริ่มจากหลักการทางคอมพิวเตอร์ สิ่งที่ควรรู้เกี่ยวกับคอมพิวเตอร์ หลักการทางด้านภาษาคอมพิวเตอร์ การเขียนแผนภูมิสายงานหรือผังงานการใช้คำสั่งต่าง ๆ ที่มีในภาษาฟอร์แทรนและตอนสุดท้ายว่าด้วยการประยุกต์ภาษาฟอร์แทรนมาใช้งาน

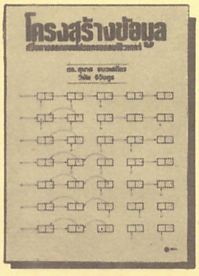
โปรแกรมคอมพิวเตอร์เบื้องต้น ภาษาโฟแทรน 4



โดย เดือน สินธุพันธ์ประทุม ราคา 49 บาท

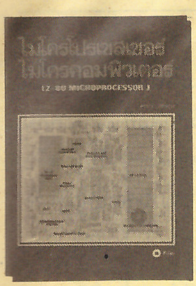
เป็นหนังสืออีกเล่มหนึ่งที่ได้รับความสนใจอย่างสูงจากผู้อ่านจนต้องพิมพ์เป็นครั้งที่ 4 เนื่องจากเนื้อหาสามารถอ่านเข้าใจได้ง่ายทั้งผู้เริ่มเรียนหรือผู้ที่พอมีพื้นความรู้มาบ้างแล้วคำสั่งพื้นฐานจากหนังสือเล่มนี้สามารถใช้ได้กับคอมพิวเตอร์ทุกเครื่องที่รับภาษาโฟแทรน 4 ได้ และยังเพิ่มเติมคำสั่งบางประเภทซึ่งใช้ได้เฉพาะเครื่องคอมพิวเตอร์ที่มีประสิทธิภาพสูงด้วยในบทที่ 10

โครงสร้างข้อมูล



โดย สุชาย ธนวเสถียร และวิชัย จิวังกูร ราคา 54 บาท

หลังจากที่เรียนภาษาคอมพิวเตอร์ตลอดจนการเขียนโปรแกรมแล้ว ขั้นต่อไปคือเรียนวิชาโครงสร้างข้อมูล วิชานี้มีความสำคัญอย่างมากเนื่องจากเป็นวิชาการพื้นฐานทางคอมพิวเตอร์ที่จะช่วยให้ผู้เขียนโปรแกรมทุกคนสามารถใช้โครงสร้างข้อมูลแบบที่เหมาะสม ซึ่งช่วยให้โปรแกรมทำงานเร็วขึ้นกินเนื้อที่น้อยลง และมีประสิทธิภาพตามที่ต้องการ ฉะนั้นถ้าท่านได้เริ่มเรียนรู้การใช้คอมพิวเตอร์โดยใช้เครื่องไมโครไม่ว่าจะเป็น APPLE จริงหรือเทียมหรือเครื่องยี่ห้อใดก็ตาม หนังสือเล่มนี้จะเป็นเล่มที่คุณพลาดไม่ได้



ไมโครโปรเซสเซอร์-ไมโครคอมพิวเตอร์

โดย ยืน ภู่วรวรรณ และวัฒนา เขียงกูล ราคา 75 บาท

หนังสือเล่มนี้เป็นการวางพื้นฐานของไมโครคอมพิวเตอร์ โดยเน้นให้เห็นถึงโครงสร้างคอมพิวเตอร์ที่จำเป็นต้องมีซีพียูหน่วยความจำ และอินพุท–เอาท์พุท นอกจากนี้ยังกล่าวถึงรายละเอียดของซีพียู Z-80 และการต่อเป็นระบบไมโครคอมพิวเตอร์ การเขียนโปรแกรม การประยุกต์ใช้งาน และสุดท้ายจะเป็นบทที่ว่าด้วยอุปกรณ์สนับสนุน ระบบไมโครคอมพิวเตอร์ที่สำคัญซึ่งได้แก่ 8214, 8212, Z-80 PIO, Z-80 CTC และ 8251



เรื่องน่ารู้เกี่ยวกับไมโครคอมพิวเตอร์

โดย ยืน ภู่วรวรรณ ราคา 53 บาท

เหมาะสำหรับผู้ที่อยากรู้รายละเอียดทั่วไปเกี่ยวกับไมโครคอมพิวเตอร์ เพื่อศึกษาข้อมูลรายละเอียดต่าง ๆ โดยเนื้อหาจะเน้นในลักษณะเรื่องทั่วไปที่เกี่ยวข้องถึงฮาร์ดแวร์ ซึ่งได้แก่ส่วนประกอบต่าง ๆ และส่วนของซอฟท์แวร์ที่ใช้กับไมโครคอมพิวเตอร์ จึงเปรียบเสมือนคู่มือที่ผู้สนใจในระดับพื้นฐานควรมีไว้ เพราะสามารถศึกษารายละเอียดต่าง ๆ รวมทั้งเป็นข้อมูลสำหรับการจัดซื้อ การพิจารณาสเปค และสร้างฐานความรู้เพื่อศึกษาไมโครคอมพิวเตอร์ในระดับที่ลึกซึ้งขึ้นไป

---

*<u>สมาชิกวารสารของซีเอ็ดทุกฉบับสั่งซื้อได้ในราคาลด 15% ค่าจัดส่งฟรี !!</u>

หาซื้อและเลือกชมได้ที่ ร้านหนังสือย่านสยามสแควร์ วังบูรพา สนามหลวงศูนย์หนังสือจุฬาฯ ร้านเมฆรุ่งเรือง ร้านเพ็ญบุญ ร้านดวงกมลสมัย

หรือสั่งซื้อทางไปรษณีย์ โดยส่งธนาณัติสั่งจ่าย ปท.ดินแดง ในนามบริษัทซีเอ็ดยูเคชั่น จำกัด 800/43–45 ซอยตระกูลสุข ถนนอโศกดินแดง เขตห้วยขวาง กทท. 10400 โทร. 2450390, 2459216

ค่าจัดส่ง (สำหรับผู้ที่ไม่ได้เป็นสมาชิก) ซื้อไม่เกิน 100 บาท ค่าจัดส่ง 7 บาท ซื้อเกิน 100 บาท ค่าจัดส่ง 7%

ซีพีเอ็ม

CP/M OPERATING SYSTEM

ซีพีเอ็มเป็นโปรแกรมจัดระบบงานที่แพร่หลายและมีผู้ใช้มากที่สุดในโลก ซีพีเอ็มเป็นโปรแกรมที่มีขีดความสามารถในการจัดระบบงานได้เป็นอย่างดี ผู้ใช้ไมโครคอมพิวเตอร์ไม่ว่าจะเป็นผู้ใช้แบบสมัครเล่น หรือใช้งานในอาชีพจะต้องเกี่ยวข้องกับการใช้ซีพีเอ็มเสมอ การใช้ซีพีเอ็มจึงเสมือนเป็นรากฐานที่ทุกคนที่สนใจไมโครคอมพิวเตอร์ควรรู้

ภายในซีพีเอ็มมีเทคนิคและรายละเอียดที่ควรรู้เป็นจำนวนมากซึ่งผู้ใช้ส่วนใหญ่มักมองข้าม หนังสือเล่มนี้จึงให้รายละเอียดเกี่ยวกับคำสั่งต่าง ๆ ที่ใช้ในซีพีเอ็มอย่างละเอียด ทั้งเป็นคำสั่งที่อยู่ภายในตัวซีพีเอ็มเองและคำสั่งประกอบที่อยู่ภายนอก

บนแผ่นซอฟต์แวร์ของซีพีเอ็มเองยังมียูทิลิตี้โปรแกรมที่สำคัญเพื่อให้ผู้ใช้ใช้งานได้ โดยเฉพาะอย่างยิ่งผู้ที่ต้องการรู้รายละเอียดเกี่ยวกับภาษาแอสเซมบลี้ หนังสือเล่มนี้จึงให้รายละเอียดเกี่ยวกับตัวเอดิเตอร์ แอสเซมเบลอร์และดีบักเกอร์ ผู้ศึกษาสามาถเรียนรู้และเข้าใจเกี่ยวกับการพัฒนาหรือเขียนโปรแกรมภาษาแอสเซมบลี้ได้เป็นอย่างดี

นอกจากนี้ภายในหนังสือยังได้ให้รายละเอียดเกี่ยวกับโครงสร้างภายในของซีพีเอ็ม การเชื่อมต่อทางซอฟต์แวร์เพื่อพัฒนาโปรแกรมหรือดัดแปลงแก้ไขโปรแกรมการจัดการระบบข้อมูลบนแผ่นดิสเก็ตต์

ISBN 974-7965-39-9